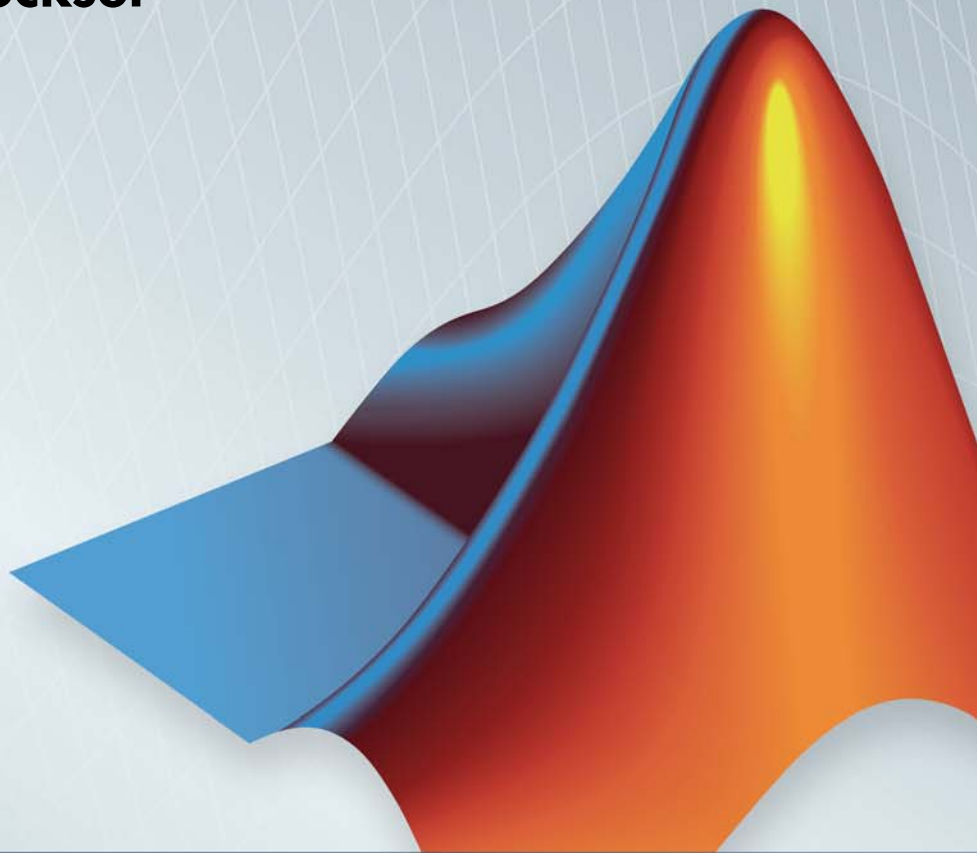


# Aerospace Blockset™

## User's Guide

**R2013a**



**MATLAB® & SIMULINK®**



## How to Contact MathWorks



[www.mathworks.com](http://www.mathworks.com) Web  
[comp.soft-sys.matlab](mailto:comp.soft-sys.matlab) Newsgroup  
[www.mathworks.com/contact\\_TS.html](http://www.mathworks.com/contact_TS.html) Technical Support



[suggest@mathworks.com](mailto:suggest@mathworks.com) Product enhancement suggestions  
[bugs@mathworks.com](mailto:bugs@mathworks.com) Bug reports  
[doc@mathworks.com](mailto:doc@mathworks.com) Documentation error reports  
[service@mathworks.com](mailto:service@mathworks.com) Order status, license renewals, passcodes  
[info@mathworks.com](mailto: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.

*Aerospace Blockset™ 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](http://www.mathworks.com/trademarks) for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

### Patents

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

## Revision History

July 2002	Online only	New for Version 1.0 (Release 13)
July 2003	Online only	Revised for Version 1.5 (Release 13SP1)
June 2004	Online only	Revised for Version 1.6 (Release 14)
October 2004	Online only	Revised for Version 1.6.1 (Release 14SP1)
March 2005	Online only	Revised for Version 1.6.2 (Release 14SP2)
May 2005	Online only	Revised for Version 2.0 (Release 14SP2+)
September 2005	First printing	Revised for Version 2.0.1 (Release 14SP3)
March 2006	Online only	Revised for Version 2.1 (Release 2006a)
September 2006	Online only	Revised for Version 2.2 (Release 2006b)
March 2007	Online only	Revised for Version 2.3 (Release 2007a)
September 2007	Second printing	Revised for Version 3.0 (Release 2007b)
March 2008	Online only	Revised for Version 3.1 (Release 2008a)
October 2008	Online only	Revised for Version 3.2 (Release 2008b)
March 2009	Online only	Revised for Version 3.3 (Release 2009a)
September 2009	Online only	Revised for Version 3.4 (Release 2009b)
March 2010	Online only	Revised for Version 3.5 (Release 2010a)
September 2010	Online only	Revised for Version 3.6 (Release 2010b)
April 2011	Online only	Revised for Version 3.7 (Release 2011a)
September 2011	Online only	Revised for Version 3.8 (Release 2011b)
March 2012	Online only	Revised for Version 3.9 (Release 2012a)
September 2012	Online only	Revised for Version 3.10 (Release 2012b)
March 2013	Online only	Revised for Version 3.11 (Release 2013a)



## Getting Started

**1**

<b>Product Description</b> .....	1-2
Key Features .....	1-2
<b>Code Generation Support</b> .....	1-3
<b>Support for Aerospace Toolbox Quaternion Functions</b> .....	1-4
<b>Related Products</b> .....	1-5
Requirements for the Aerospace Blockset Product .....	1-5
Other Related Products .....	1-5
<b>Explore the NASA HL-20 Model</b> .....	1-6
Introduction .....	1-6
What This Example Illustrates .....	1-6
Open the Model .....	1-7
Key Subsystems .....	1-10
NASA HL-20 Example .....	1-11
Modify the Model .....	1-13
<b>Open Aerospace Examples</b> .....	1-16

## Aerospace Blockset Software

**2**

<b>Create Aerospace Models</b> .....	2-2
Basic Steps .....	2-2
<b>Build a Simple Actuator System</b> .....	2-5

Build the Model .....	2-5
Run the Simulation .....	2-10
<b>About Aerospace Coordinate Systems .....</b>	<b>2-12</b>
Fundamental Coordinate System Concepts .....	2-12
Coordinate Systems for Modeling .....	2-14
Coordinate Systems for Navigation .....	2-16
Coordinate Systems for Display .....	2-19
References .....	2-20
<b>Flight Simulator Interface .....</b>	<b>2-22</b>
About the FlightGear Interface .....	2-22
Obtain FlightGear .....	2-22
Configure Your Computer for FlightGear .....	2-23
Install and Start FlightGear .....	2-26
<b>Work with the Flight Simulator Interface .....</b>	<b>2-28</b>
Introduction .....	2-28
About Aircraft Geometry Models .....	2-29
Work with Aircraft Geometry Models .....	2-31
Run FlightGear with the Simulink Models .....	2-34
Run the NASA HL-20 Example with FlightGear .....	2-42
Send and Receive Data .....	2-45

## Case Studies

### 3

<b>Ideal Airspeed Correction .....</b>	<b>3-2</b>
Introduction .....	3-2
Airspeed Correction Models .....	3-2
Measure Airspeed .....	3-4
Model Airspeed Correction .....	3-5
Simulate Airspeed Correction .....	3-7
<b>1903 Wright Flyer .....</b>	<b>3-10</b>
Introduction .....	3-10
Wright Flyer Model .....	3-11
Airframe Subsystem .....	3-11
Environment Subsystem .....	3-15

Pilot Subsystem .....	3-16
Run the Simulation .....	3-17
References .....	3-19
<b>NASA HL-20 Lifting Body Airframe .....</b>	<b>3-20</b>
Introduction .....	3-20
NASA HL-20 Lifting Body .....	3-20
The HL-20 Airframe and Controller Model .....	3-21
References .....	3-34

## Blocks — Alphabetical List

**4** |

## Aerospace Units

**A** |

## Index





# Getting Started

---

- “Product Description” on page 1-2
- “Code Generation Support” on page 1-3
- “Support for Aerospace Toolbox Quaternion Functions” on page 1-4
- “Related Products” on page 1-5
- “Explore the NASA HL-20 Model” on page 1-6
- “Open Aerospace Examples” on page 1-16

## Product Description

### **Model and simulate aircraft, spacecraft, and propulsion systems**

Aerospace Blockset™ software extends Simulink® with blocks for modeling and simulating aircraft, spacecraft, rocket, and propulsion systems, as well as unmanned airborne vehicles. It also includes blocks that implement mathematical representations from aerospace standards, common references, and first principles. Blocks for modeling equations of motion and for navigation, gain scheduling, visualization, unit conversion, and other key operations are also provided.

### **Key Features**

- Simulates aerospace vehicle components, including propulsion systems, control systems, mass properties, and actuators
- Models flight dynamics, including three- and six-degrees-of-freedom equations of motion with fixed or variable mass
- Provides options for visualizing vehicle dynamics in a three-dimensional environment, including an interface to FlightGear flight simulator
- Includes standards-based environmental models for atmosphere, gravity, wind, geoid height, and magnetic field
- Implements predefined utilities for converting units, transforming coordinate systems and spatial representations, and performing common aerospace math operations

## **Code Generation Support**

Use the Aerospace Blockset software with the Simulink Coder™ software to automatically generate code for real-time execution in rapid prototyping and for hardware-in-the-loop systems.

## Support for Aerospace Toolbox Quaternion Functions

The Aerospace Blockset product supports the following Aerospace Toolbox quaternion functions in the MATLAB Function block:

```
quatconj  
quatinv  
quatmod  
quatmultiply  
quatdivide  
quatnorm  
quatnormalize
```

For further information on using the MATLAB Function block, see:

- “What Is a MATLAB Function Block?”
- `asbQuatEML` example, which illustrates quaternions and models the equations

## Related Products

In this section...
“Requirements for the Aerospace Blockset Product” on page 1-5
“Other Related Products” on page 1-5

### Requirements for the Aerospace Blockset Product

In particular, the Aerospace Blockset product requires current versions of these products:

- MATLAB
- Aerospace Toolbox
- Simulink

### Other Related Products

The related products listed on the Aerospace Blockset product page at the MathWorks Web site include toolboxes and blocksets that extend the capabilities of the MATLAB® and Simulink products. These products will enhance your use of the Aerospace Blockset product in various applications.

### For More Information About MathWorks Products

For more information about any MathWorks® software products, see either

- The online documentation for that product if it is installed
- The MathWorks Web site at [www.mathworks.com](http://www.mathworks.com); see the “Products” section

## Explore the NASA HL-20 Model

In this section...
“Introduction” on page 1-6
“What This Example Illustrates” on page 1-6
“Open the Model” on page 1-7
“Key Subsystems” on page 1-10
“NASA HL-20 Example” on page 1-11
“Modify the Model” on page 1-13

### Introduction

This section introduces a NASA HL-20 lifting body airframe model that uses blocks from the Aerospace Blockset software to simulate the airframe of a NASA HL-20 lifting body, in conjunction with other Simulink blocks.

The model simulates the NASA HL-20 lifting body airframe approach and landing flight phases using an automatic-landing controller.

For more information on this model, see “NASA HL-20 Lifting Body Airframe” on page 3-20.

### What This Example Illustrates

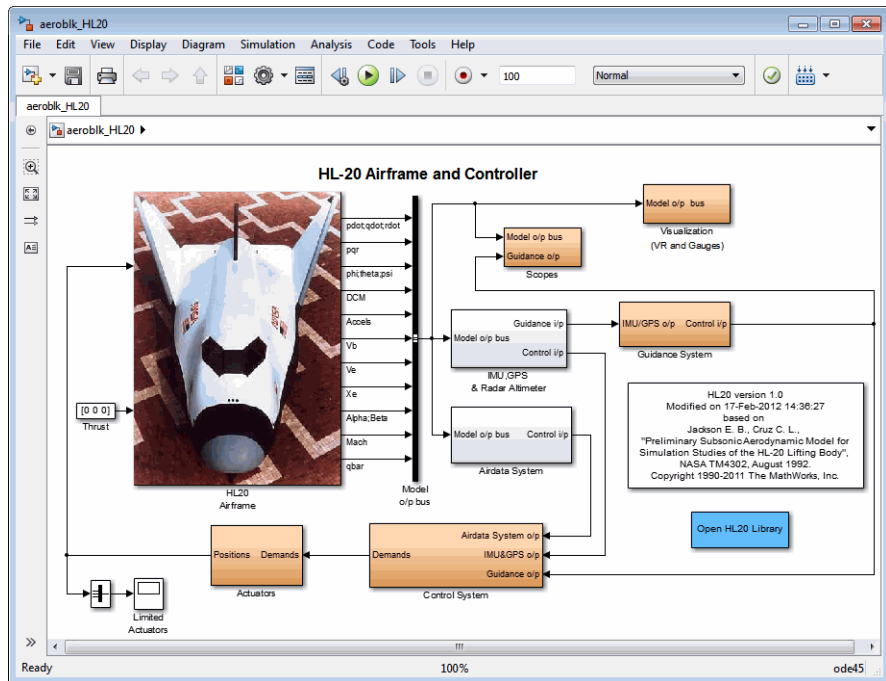
The NASA HL-20 lifting body airframe example illustrates the following features of the blockset:

- Representing bodies and degrees of freedom with the Equations of Motion library blocks
- Using the Aerospace Blockset blocks with other Simulink blocks
- Feeding in and feeding out Simulink signals to and from Aerospace Blockset blocks with Actuator and Sensor blocks
- Encapsulating groups of blocks into subsystems

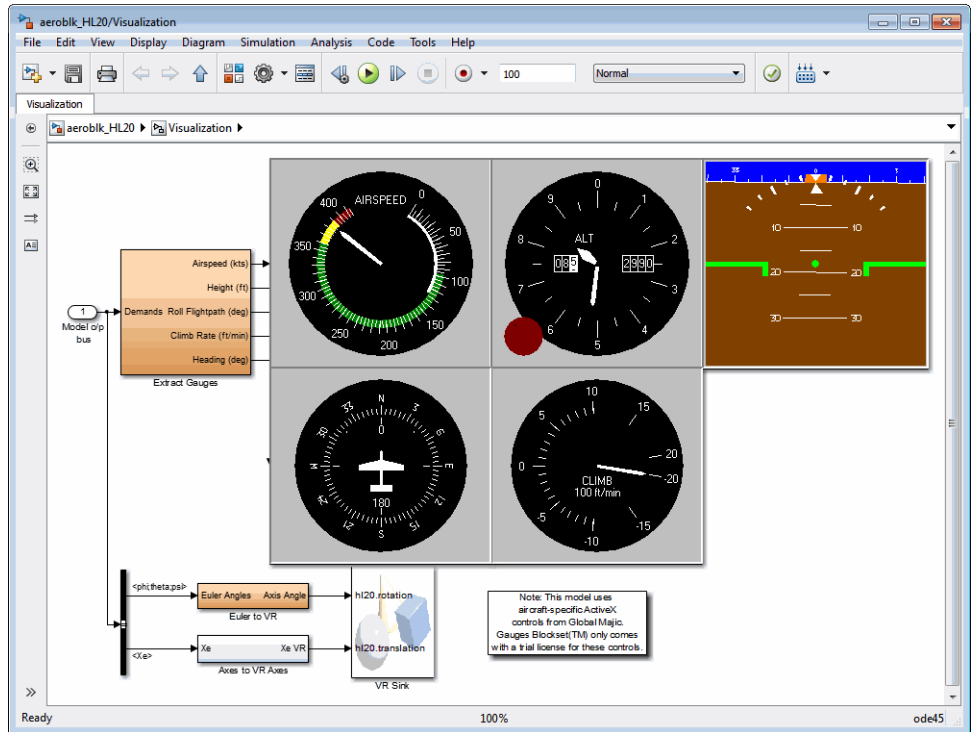
- Visualizing an aircraft with Simulink 3D Animation™ and Gauges Blockset™ library blocks.

## Open the Model

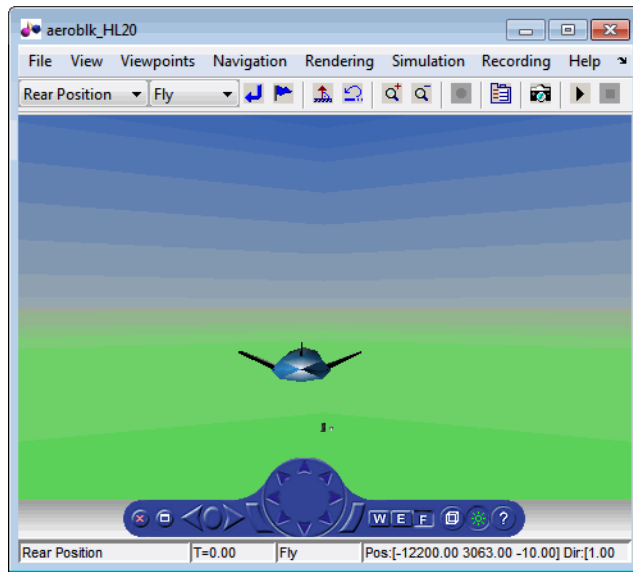
To open the NASA HL-20 airframe example, type the example name, `aeroblk_HL20`, at the MATLAB command line. The model opens.

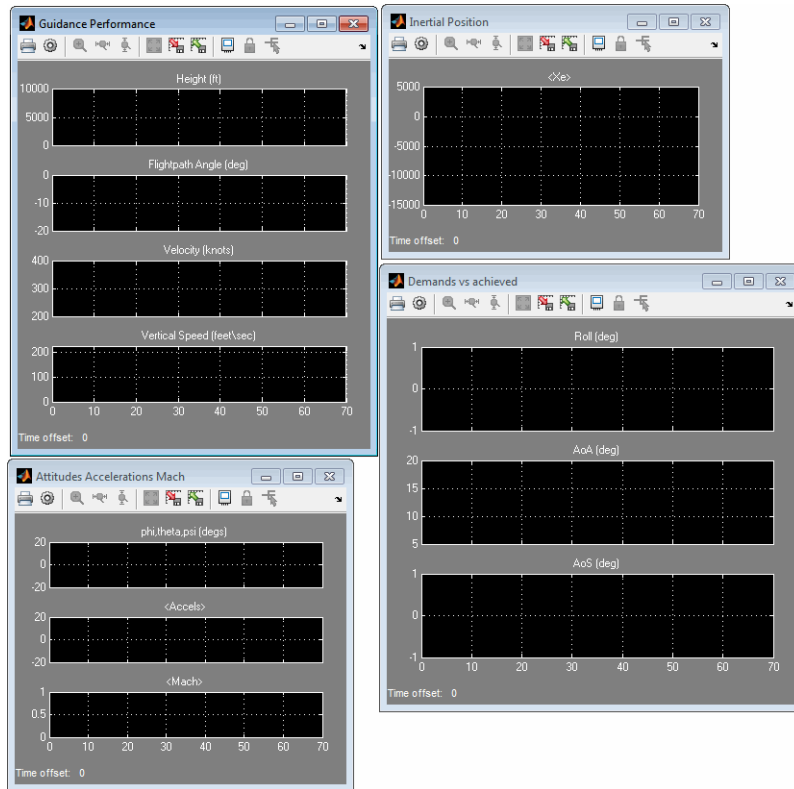


The visualization subsystem, four scopes, and a Simulink 3D Animation viewer for the airframe also appear.









## Key Subsystems

The model implements the airframe using the following subsystems:

- The 6DoF (Euler Angles) subsystem implements the 6DoF (Euler Angles) block along with other Simulink blocks.
- The Environment Models subsystem implements the WGS84 Gravity Model and COESA Atmosphere Model blocks. It also contains a Wind Models subsystem that implements a number of wind blocks.
- The Alpha, Beta, Mach subsystem implements the Incidence, Sideslip & Airspeed, Mach Number, and Dynamic Pressure blocks. These blocks calculate aerodynamic coefficient values and lookup functionality.

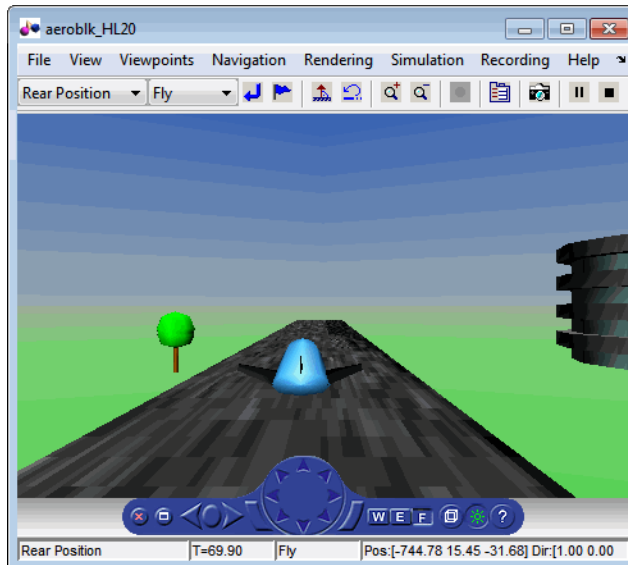
- The Forces and Moments subsystem implements the Aerodynamic Forces and Moments block. This subsystem calculates body forces and body moments.
- The Aerodynamic Coefficients subsystem implements several subsystems to calculate six aerodynamic coefficients.

## NASA HL-20 Example

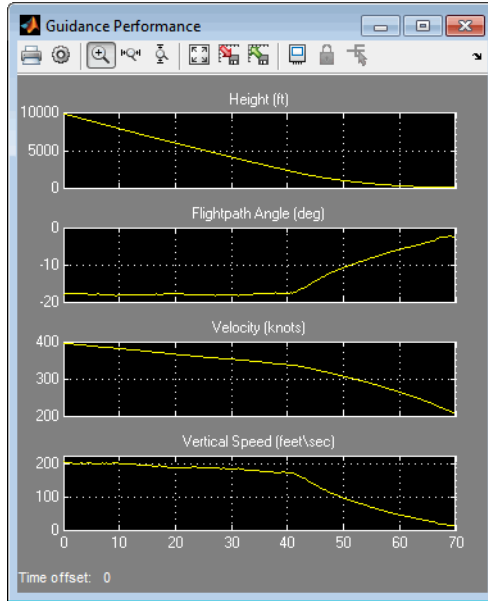
Running an example lets you observe the model simulation in real time. After you run the example, you can examine the resulting data in plots, graphs, and other visualization tools. To run this model, follow these steps:

- 1 If it is not already open, open the aeroblk\_HL20 example.
- 2 From the **Simulation** menu, select **Run**. On Microsoft® Windows® systems, you can also click the **Run** button in the model window toolbar.

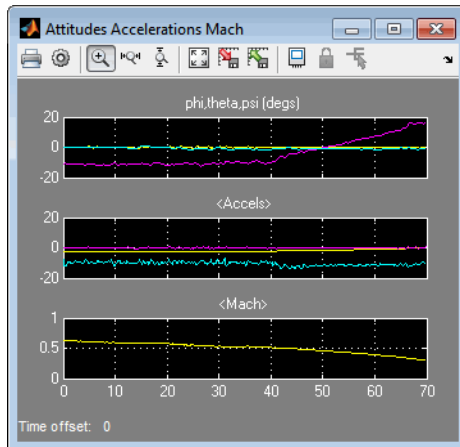
The simulation proceeds until the aircraft lands:



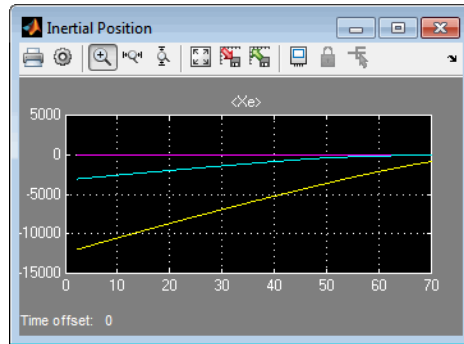
**View of the landed airframe**



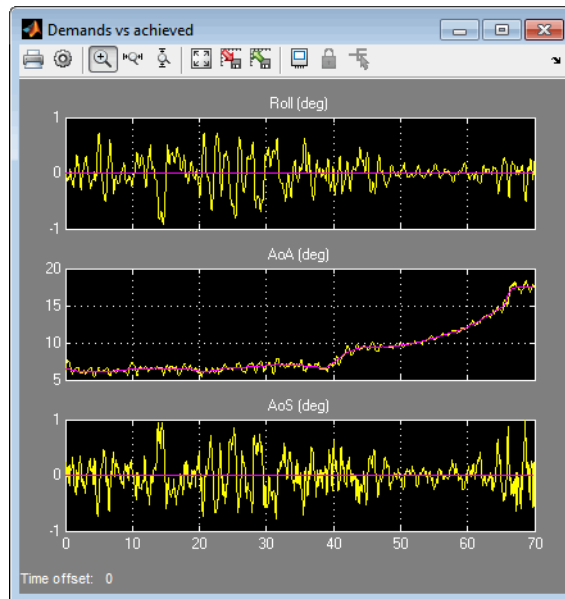
**Plot that Measures Guidance Performance**



**Plot that Measures Altitude Accelerations Mach**



**Plot that Measures Inertial Position**



**Plot that Measures Demand Data Against Achieved Data**

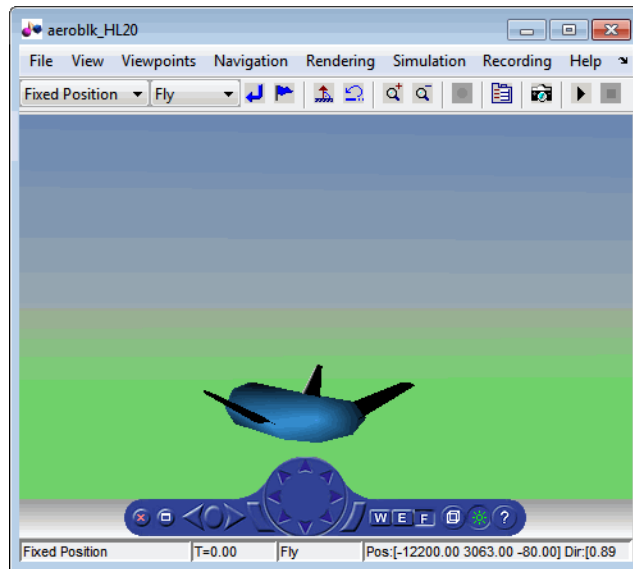
## Modify the Model

You can adjust the airframe model settings and examine the effects on simulation performance. Here is one modification that you can try. It changes the camera point of view for the landing animation.

## Change the Animation Point of View

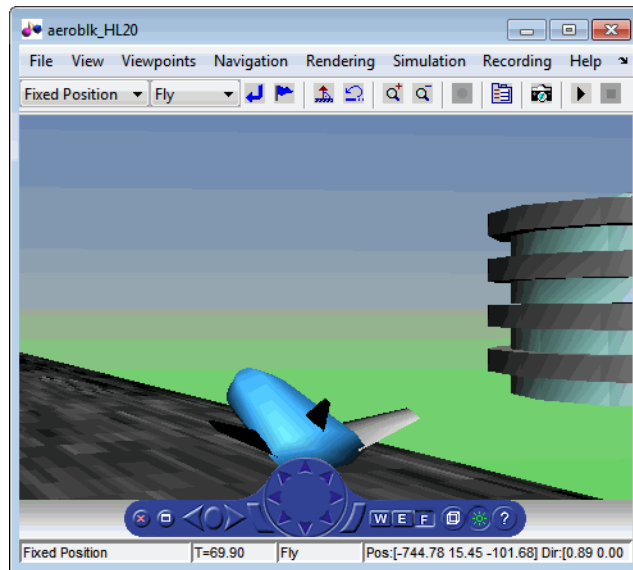
By default, the airframe animation viewpoint is Rear position, which means the view tracks with the airframe flight path from the rear. You can change the animation point of view by selecting another viewpoint from the Simulink 3D Animation viewer:

- 1 Open the aeroblk\_HL20 model, and click the Simulink 3D Animation viewer.
- 2 From the list of existing viewpoints, change the viewpoint to Fixed Position.



The airframe view changes to a fixed position.

- 3 Start the model again. Notice the different airframe viewpoint when the airframe lands.



You can experiment with different viewpoints to watch the animation from different perspectives.

## Open Aerospace Examples

To open an Aerospace Blockset example from Help Browser:

- 1** Open the MATLAB Command Window.
- 2** Click the question mark.
- 3** Navigate to Aerospace Blockset and click **Examples** in the top right-hand corner.

The Aerospace Blockset Examples page is displayed.

Alternatively, you can open the **Examples** window by entering `demodemos` at the MATLAB command line.

For in-depth case studies of the following examples, see:

- “Ideal Airspeed Correction” on page 3-2
- “1903 Wright Flyer” on page 3-10
- “NASA HL-20 Lifting Body Airframe” on page 3-20



# Aerospace Blockset Software

---

- “Create Aerospace Models” on page 2-2
- “Build a Simple Actuator System” on page 2-5
- “About Aerospace Coordinate Systems” on page 2-12
- “Flight Simulator Interface” on page 2-22
- “Work with the Flight Simulator Interface” on page 2-28

## Create Aerospace Models

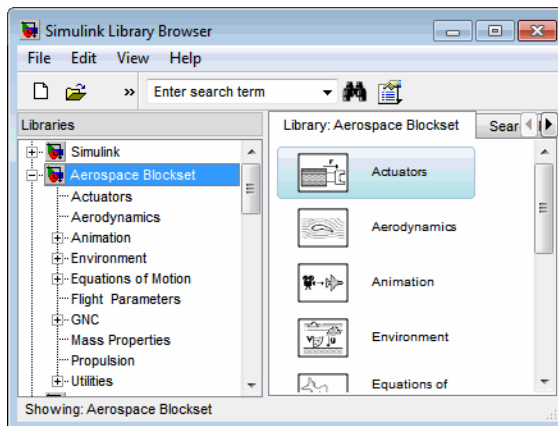
### Basic Steps

Regardless of the model's complexity, you use the same essential steps for creating an aerospace model as you would for creating any other Simulink model.

- 1 Open the Aerospace Blockset Library.

Type `simulink` to open Simulink Library Browser in the MATLAB

Command Window or click the  button. The left pane contains a list of all the blocksets that you currently have installed.

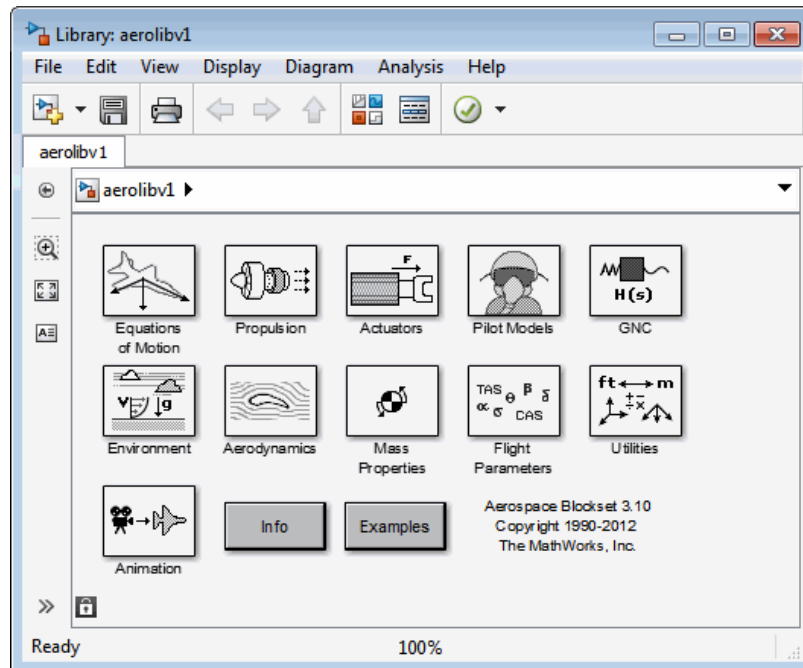


Navigate to the Aerospace Blockset node and expand the hierarchical list to display that blockset's libraries within the browser.

Alternatively, to open the Aerospace Blockset window from the MATLAB command line, enter

```
aerolib
```

Double-click any library in the window to display its contents. The following figure shows the Aerospace Blockset library window.



- 2** *Select and position the blocks.* You must first select the blocks that you need to build your model, and then position the blocks in the model window. For the majority of Simulink models, you select one or more blocks from each of the following categories:
- a** Source blocks generate or import signals into the model, such as a sine wave, a clock, or limited-band white noise.
  - b** Simulation blocks can consist of almost any type of block that performs an action in the simulation. A simulation block represents a part of the model functionality to be simulated, such as an actuator block, a mathematical operation, a block from the Aerospace Blockset library, and so on.
  - c** Signal Routing blocks route signals from one point in a model to another. If you need to combine or redirect two or more signals in your model, you will probably use a Simulink Signal Routing block, such as Mux and Demux.

As an alternative to the Mux block, you can use the **Vector** option of the Concatenate block **Mode** parameter (also known as the Vector Concatenate block). This block provides a more general way for you to route signals from one point in the a model to another. The Vector mode takes as input a vector of signals of the same data type and creates a contiguous output signal. Depending on the input, this block outputs a row or column vector if any of the inputs are row or column vectors, respectively.

- d** Sink blocks display, write, or save model output. To see the results of the simulation, you must use a Sink block.
- 3** *Configure the blocks.* Most blocks feature configuration options that let you customize block functionality to specific simulation parameters. For example, the ISA Atmosphere Model block provides configuration options for setting the height of the troposphere, tropopause, and air density at sea level.
- 4** *Connect the blocks.* To create signal pathways between blocks, you connect the blocks to each other. You can do this manually by clicking and dragging, or you can connect blocks automatically.
- 5** *Encapsulate subsystems.* Systems made with Aerospace Blockset blocks can function as subsystems of larger, more complex models, like subsystems in any Simulink model.

## Build a Simple Actuator System

### In this section...

“Build the Model” on page 2-5

“Run the Simulation” on page 2-10

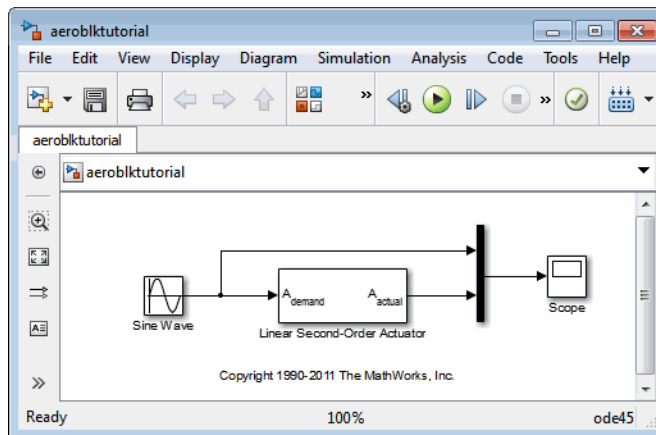
### Build the Model

The Simulink product is a software environment for modeling, simulating, and analyzing dynamic systems. Try building a simple model that drives an actuator with a sine wave and displays the actuator’s position superimposed on the sine wave.

---

**Note** If you prefer to open the complete model shown below instead of building it, enter `aeroblktutorial` at the MATLAB command line.

---




The following section (“Create a Model” on page 2-6) explains how to build a model on Windows platforms. You can use this same procedure to build a model on UNIX® platforms.

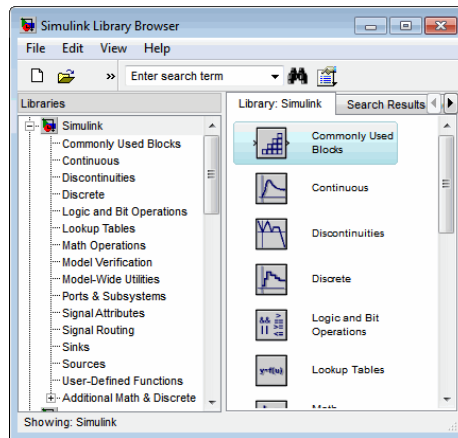
The section describes how to build the model. It does not describe how to set the configuration parameters for the model. See “Configure Simulation”. That

topic describes setting configuration parameters for models. If you do not set any configuration parameters, simulating models might cause warnings like:


```
Warning: Using a default value of 0.2 for maximum step size.
The simulation step size will be equal to or less than this
value. You can disable this diagnostic by setting
'Automatic solver parameter selection' diagnostic to 'none'
in the Diagnostics page of the configuration parameters
dialog
```

### Create a Model

- 1 Click the  button in the MATLAB Toolstrip or enter `simulink` at the MATLAB command line. The Simulink library browser appears.



- 2 Select **New > Model** from the **File** menu in the Library Browser. A new model window appears on your screen.
- 3 Add a Sine Wave block to the model.
  - a Click **Sources** in the Library Browser to view the blocks in the Simulink Sources library.
  - b Drag the Sine Wave block from the Sources library into the new model window.

- 4** Add a Linear Second-Order Actuator block to the model.
  - a** Click the  symbol next to **Aerospace Blockset** in the Library Browser to expand the hierarchical list of the aerospace blocks.
  - b** In the expanded list, click **Actuators** to view the blocks in the Actuator library.
  - c** Drag the Linear Second-Order Actuator block into the model window.
- 5** Add a Mux block to the model.
  - a** Click **Signal Routing** in the Library Browser to view the blocks in the Simulink Signals & Systems library.
  - b** Drag the Mux block from the Signal Routing library into the model window.
- 6** Add a Scope block to the model.
  - a** Click **Sinks** in the Library Browser to view the blocks in the Simulink Sinks library.
  - b** Drag the Scope block from the Sinks library into the model window.
- 7** Resize the Mux block in the model.
  - a** Click the Mux block to select the block.
  - b** Hold down the mouse button and drag a corner of the Mux block to change the size of the block.
- 8** Connect the blocks.
  - a** Position the pointer near the output port of the Sine Wave block. Hold down the mouse button and drag the line that appears until it touches the input port of the Linear Second-Order Actuator block. Release the mouse button.
  - b** Using the same technique, connect the output of the Linear Second-Order Actuator block to the second input port of the Mux block.
  - c** Using the same technique, connect the output of the Mux block to the input port of the Scope block.
  - d** Position the pointer near the first input port of the Mux block. Hold down the mouse button and drag the line that appears over the line

from the output port of the Sine Wave block until double crosshairs appear. Release the mouse button. The lines are connected when a knot is present at their intersection.

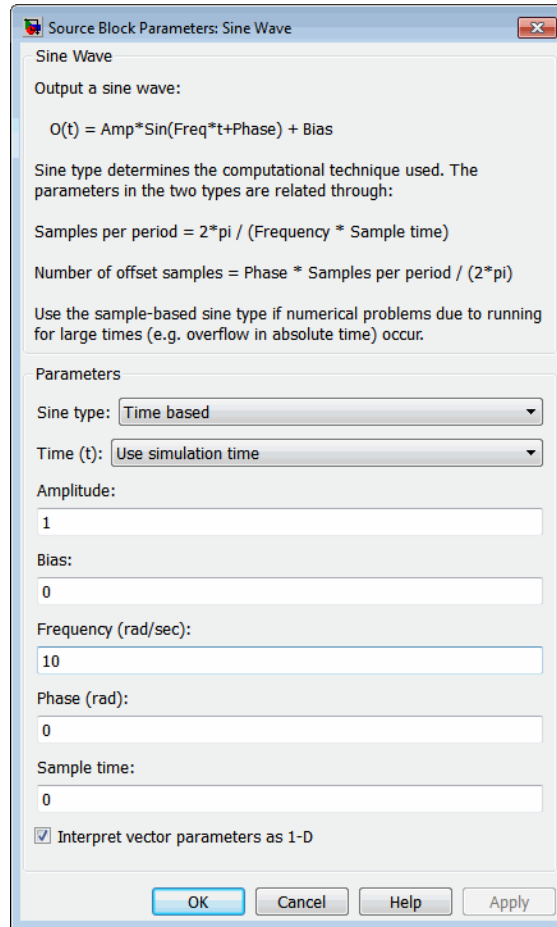
9 Set the block parameters.

- a Double-click the Sine Wave block. The dialog box that appears allows you to set the block's parameters.

For this example, configure the block to generate a 10 rad/s sine wave by entering 10 for the **Frequency** parameter. The sinusoid has the default amplitude of 1 and phase of 0 specified by the **Amplitude** and **Phase offset** parameters.

- b Click **OK**.

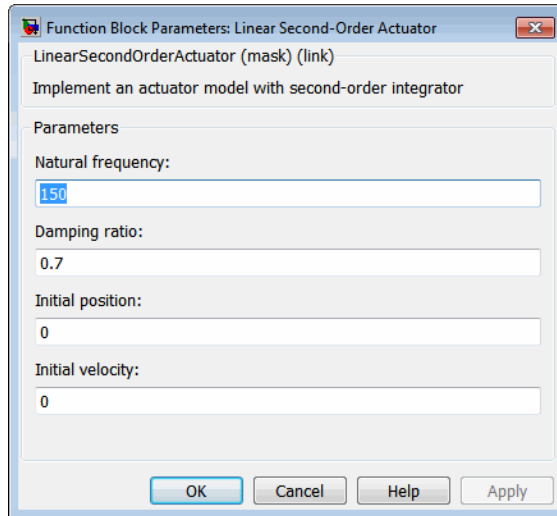




- c Double-click the Linear Second-Order Actuator block.

In this example, the actuator has the default natural frequency of 150 rad/s, a damping ratio of 0.7, and an initial position of 0 radians specified by the **Natural frequency**, **Damping ratio**, and **Initial position** parameters.

- d Click **OK**.



### Run the Simulation

You can now run the model that you built to see how the system behaves in time:

- 1 Double-click the Scope block if the Scope window is not already open on your screen. The Scope window appears.
- 2 Select **Run** from the **Simulation** menu in the model window. The signal containing the 10 rad/s sinusoid and the signal containing the actuator position are plotted on the scope.
- 3 Adjust the Scope block's display. While the simulation is running, right-click the *y*-axis of the scope and select **Autoscale**. The vertical range of the scope is adjusted to better fit the signal.
- 4 Vary the Sine Wave block parameters.
  - a While the simulation is running, double-click the Sine Wave block to open its parameter dialog box.
  - b You can then change the frequency of the sinusoid. Try entering 1 or 20 in the **Frequency** field. Close the Sine Wave dialog box to enter your change. You can then observe the changes on the scope.

**5** Select **Stop** from the **Simulation** menu to stop the simulation.

Many parameters *cannot* be changed while a simulation is running. This is usually the case for parameters that directly or indirectly alter a signal's dimensions or sample rate. However, there are some parameters, like the Sine Wave **Frequency** parameter, that you can *tune* without stopping the simulation.

### **Run a Simulation from a Script**

You can also modify and run a Simulink simulation from a script. By doing this, you can automate the variation of model parameters to explore a large number of simulation conditions rapidly and efficiently. For information on how to do this, see “About Programmatic Simulation”.

## About Aerospace Coordinate Systems

In this section...
“Fundamental Coordinate System Concepts” on page 2-12
“Coordinate Systems for Modeling” on page 2-14
“Coordinate Systems for Navigation” on page 2-16
“Coordinate Systems for Display” on page 2-19
“References” on page 2-20

### Fundamental Coordinate System Concepts

Coordinate systems allow you to keep track of an aircraft or spacecraft’s position and orientation in space. The Aerospace Blockset coordinate systems are based on these underlying concepts from geodesy, astronomy, and physics.

#### Definitions

The blockset uses *right-handed* (RH) *Cartesian* coordinate systems. The *right-hand rule* establishes the *x-y-z* sequence of coordinate axes.

An *inertial frame* is a nonaccelerating motion reference frame. In an inertial frame, Newton’s second law holds:  $\text{force} = \text{mass} \times \text{acceleration}$ . Loosely speaking, acceleration is defined with respect to the distant cosmos, and an inertial frame is often said to be nonaccelerated with respect to the “fixed stars.” Because the Earth and stars move so slowly with respect to one another, this assumption is a very accurate approximation.

Strictly defined, an inertial frame is a member of the set of all frames not accelerating relative to one another. A *noninertial frame* is any frame accelerating relative to an inertial frame. Its acceleration, in general, includes both translational and rotational components, resulting in *pseudoforces* (*pseudogravity*, as well as *Coriolis* and *centrifugal forces*).

The blockset models the Earth’s shape (the *geoid*) as an oblate spheroid, a special type of ellipsoid with two longer axes equal (defining the *equatorial plane*) and a third, slightly shorter (*geopolar*) axis of symmetry. The equator is the intersection of the equatorial plane and the Earth’s surface. The

geographic poles are the intersection of the Earth's surface and the geopolar axis. In general, the Earth's geopolar and rotation axes are not identical.

Latitudes parallel the equator. Longitudes parallel the geopolar axis. The *zero longitude* or *prime meridian* passes through Greenwich, England.

## Approximations

The blockset makes three standard approximations in defining coordinate systems relative to the Earth.

- The Earth's surface or geoid is an oblate spheroid, defined by its longer equatorial and shorter geopolar axes. In reality, the Earth is slightly deformed with respect to the standard geoid.
- The Earth's rotation axis and equatorial plane are perpendicular, so that the rotation and geopolar axes are identical. In reality, these axes are slightly misaligned, and the equatorial plane wobbles as the Earth rotates. This effect is negligible in most applications.
- The only noninertial effect in Earth-fixed coordinates is due to the Earth's rotation about its axis. This is a *rotating, geocentric* system. The blockset ignores the Earth's acceleration around the Sun, the Sun's acceleration in the Galaxy, and the Galaxy's acceleration through the cosmos. In most applications, only the Earth's rotation matters.

This approximation must be changed for spacecraft sent into deep space, i.e., outside the Earth-Moon system, and a heliocentric system is preferred.

## Motion with Respect to Other Planets

The blockset uses the standard WGS-84 geoid to model the Earth. You can change the equatorial axis length, the flattening, and the rotation rate.

You can represent the motion of spacecraft with respect to any celestial body that is well approximated by an oblate spheroid by changing the spheroid size, flattening, and rotation rate. If the celestial body is rotating westward (retrogradely), make the rotation rate negative.

## Coordinate Systems for Modeling

Modeling aircraft and spacecraft is simplest if you use a coordinate system fixed in the body itself. In the case of aircraft, the forward direction is modified by the presence of wind, and the craft's motion through the air is not the same as its motion relative to the ground.

See “Equations of Motion” for further details on how the Aerospace Blockset product implements body and wind coordinates.

### Body Coordinates

The noninertial body coordinate system is fixed in both origin and orientation to the moving craft. The craft is assumed to be rigid.

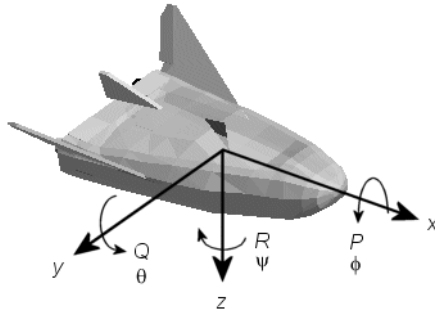
The orientation of the body coordinate axes is fixed in the shape of body.

- The  $x$ -axis points through the nose of the craft.
- The  $y$ -axis points to the right of the  $x$ -axis (facing in the pilot's direction of view), perpendicular to the  $x$ -axis.
- The  $z$ -axis points down through the bottom the craft, perpendicular to the  $xy$  plane and satisfying the RH rule.

**Translational Degrees of Freedom.** Translations are defined by moving along these axes by distances  $x$ ,  $y$ , and  $z$  from the origin.

**Rotational Degrees of Freedom.** Rotations are defined by the Euler angles  $P$ ,  $Q$ ,  $R$  or  $\Phi$ ,  $\Theta$ ,  $\Psi$ . They are:

$P$ or $\Phi$	Roll about the $x$ -axis
$Q$ or $\Theta$	Pitch about the $y$ -axis
$R$ or $\Psi$	Yaw about the $z$ -axis



## Wind Coordinates

The noninertial wind coordinate system has its origin fixed in the rigid aircraft. The coordinate system orientation is defined relative to the craft's velocity  $V$ .

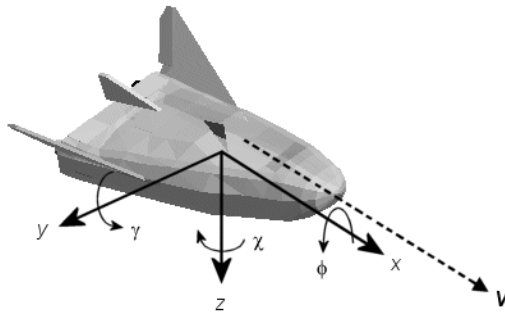
The orientation of the wind coordinate axes is fixed by the velocity  $V$ .

- The  $x$ -axis points in the direction of  $V$ .
- The  $y$ -axis points to the right of the  $x$ -axis (facing in the direction of  $V$ ), perpendicular to the  $x$ -axis.
- The  $z$ -axis points perpendicular to the  $xy$  plane in whatever way needed to satisfy the RH rule with respect to the  $x$ - and  $y$ -axes.

**Translational Degrees of Freedom.** Translations are defined by moving along these axes by distances  $x$ ,  $y$ , and  $z$  from the origin.

**Rotational Degrees of Freedom.** Rotations are defined by the Euler angles  $\Phi$ ,  $\gamma$ ,  $\chi$ . They are:

$\Phi$	Bank angle about the $x$ -axis
$\gamma$	Flight path about the $y$ -axis
$\chi$	Heading angle about the $z$ -axis



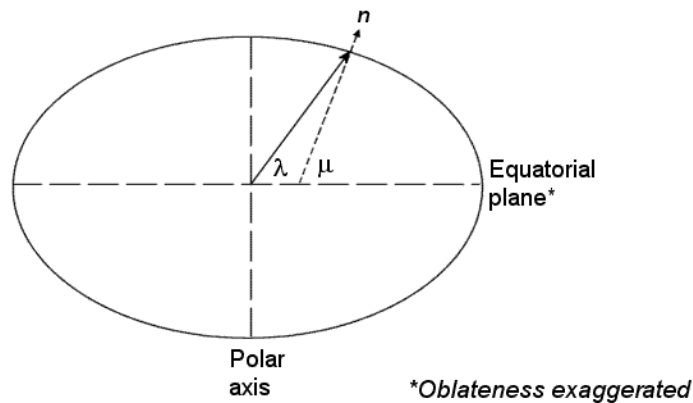
## Coordinate Systems for Navigation

Modeling aerospace trajectories requires positioning and orienting the aircraft or spacecraft with respect to the rotating Earth. Navigation coordinates are defined with respect to the center and surface of the Earth.

### Geocentric and Geodetic Latitudes

The *geocentric latitude*  $\lambda$  on the Earth's surface is defined by the angle subtended by the radius vector from the Earth's center to the surface point with the equatorial plane.

The *geodetic latitude*  $\mu$  on the Earth's surface is defined by the angle subtended by the surface normal vector  $n$  and the equatorial plane.



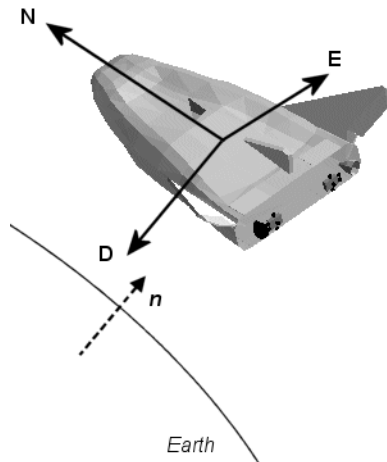


## NED Coordinates

The north-east-down (NED) system is a noninertial system with its origin fixed at the aircraft or spacecraft's center of gravity. Its axes are oriented along the geodetic directions defined by the Earth's surface.

- The  $x$ -axis points north parallel to the geoid surface, in the polar direction.
- The  $y$ -axis points east parallel to the geoid surface, along a latitude curve.
- The  $z$ -axis points downward, toward the Earth's surface, antiparallel to the surface's outward normal  $n$ .

Flying at a constant altitude means flying at a constant  $z$  above the Earth's surface.

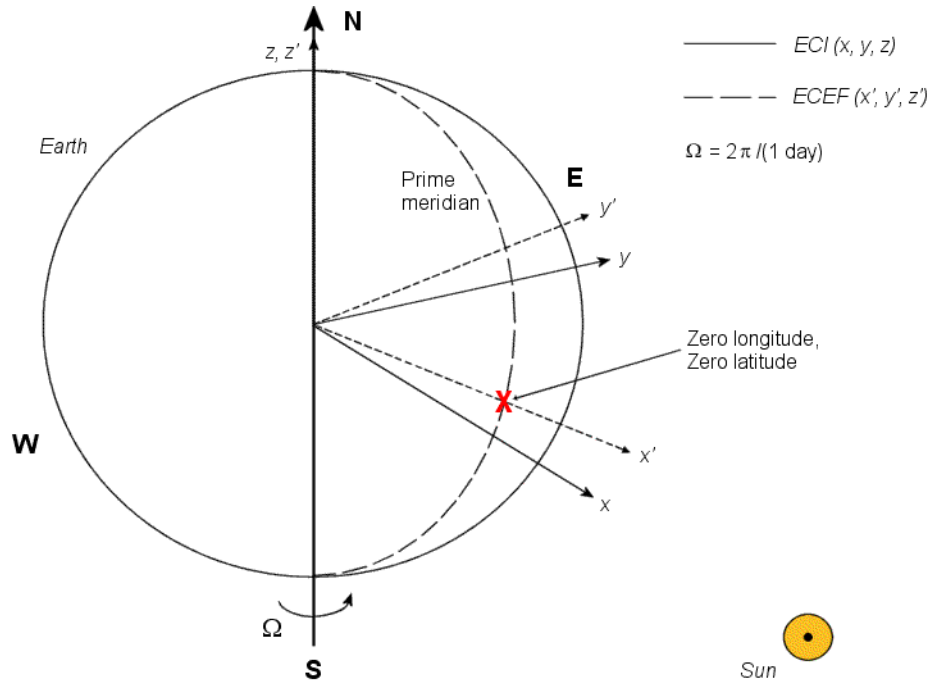


## ECI Coordinates

The Earth-centered inertial (ECI) system is a mixed inertial system. It is oriented with respect to the Sun. Its origin is fixed at the center of the Earth. (See figure following.)

- The  $z$ -axis points northward along the Earth's rotation axis.
- The  $x$ -axis points outward in the Earth's equatorial plane exactly at the Sun. (This rule ignores the Sun's oblique angle to the equator, which varies with season. The actual Sun always remains in the  $xz$  plane.)

- The  $y$ -axis points into the eastward quadrant, perpendicular to the  $xz$  plane so as to satisfy the RH rule.



### Earth-Centered Coordinates

#### ECEF Coordinates

The Earth-center, Earth-fixed (ECEF) system is a noninertial system that rotates with the Earth. Its origin is fixed at the center of the Earth. (See figure preceding.)

- The  $z'$ -axis points northward along the Earth's rotation axis.
- The  $x'$ -axis points outward along the intersection of the Earth's equatorial plane and prime meridian.
- The  $y'$ -axis points into the eastward quadrant, perpendicular to the  $x$ - $z$  plane so as to satisfy the RH rule.

## Coordinate Systems for Display

Several display tools are available for use with the Aerospace Blockset product. Each has a specific coordinate system for rendering motion.

### **MATLAB Graphics Coordinates**

See the “Coordinate System” for more information about the MATLAB Graphics coordinate axes.

MATLAB Graphics uses this default coordinate axis orientation:

- The  $x$ -axis points out of the screen.
- The  $y$ -axis points to the right.
- The  $z$ -axis points up.

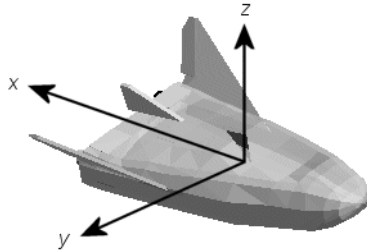
### **FlightGear Coordinates**

FlightGear is an open-source, third-party flight simulator with an interface supported by the blockset.

- “Work with the Flight Simulator Interface” on page 2-28 discusses the blockset interface to FlightGear.
- See the FlightGear documentation at [www.flightgear.org](http://www.flightgear.org) for complete information about this flight simulator.

The FlightGear coordinates form a special body-fixed system, rotated from the standard body coordinate system about the  $y$ -axis by -180 degrees:

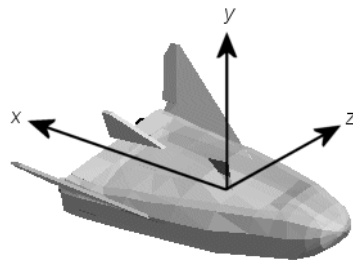
- The  $x$ -axis is positive toward the back of the vehicle.
- The  $y$ -axis is positive toward the right of the vehicle.
- The  $z$ -axis is positive upward, e.g., wheels typically have the lowest  $z$  values.



### AC3D Coordinates

AC3D is a low-cost, widely used, geometry editor available from [www.ac3d.org](http://www.ac3d.org). Its body-fixed coordinates are formed by inverting the three standard body coordinate axes:

- The  $x$ -axis is positive toward the back of the vehicle.
- The  $y$ -axis is positive upward, e.g., wheels typically have the lowest  $y$  values.
- The  $z$ -axis is positive to the left of the vehicle.



### References

*Recommended Practice for Atmospheric and Space Flight Vehicle Coordinate Systems*, R-004-1992, ANSI/AIAA, February 1992.

Mapping Toolbox documentation, The MathWorks, Inc., Natick, Massachusetts.  
<http://www.mathworks.com/help/toolbox/map/index.html>.

Rogers, R. M., *Applied Mathematics in Integrated Navigation Systems*, AIAA, Reston, Virginia, 2000.

Sobel, D., *Longitude*, Walker & Company, New York, 1995.

Stevens, B. L., and F. L. Lewis, *Aircraft Control and Simulation*, 2nd ed., *Aircraft Control and Simulation*, Wiley-Interscience, New York, 2003.

Thomson, W. T., *Introduction to Space Dynamics*, John Wiley & Sons, New York, 1961/Dover Publications, Mineola, New York, 1986.

World Geodetic System 1984 (WGS 84),  
<http://earth-info.nga.mil/GandG/wgs84/>.

## Flight Simulator Interface

In this section...
“About the FlightGear Interface” on page 2-22
“Obtain FlightGear” on page 2-22
“Configure Your Computer for FlightGear” on page 2-23
“Install and Start FlightGear” on page 2-26

### About the FlightGear Interface

The Aerospace Blockset product supports an interface to the third-party FlightGear flight simulator, an open source software package available through a GNU General Public License (GPL). The FlightGear flight simulator interface included with the blockset is a unidirectional transmission link from the Simulink interface to FlightGear using FlightGear’s published `net_fdm` binary data exchange protocol. Data is transmitted via UDP network packets to a running instance of FlightGear. The blockset supports multiple standard binary distributions of FlightGear. See “Run FlightGear with the Simulink Models” on page 2-34 for interface details.

FlightGear is a separate software entity neither created, owned, nor maintained by MathWorks.

- To report bugs in or request enhancements to the Aerospace Blockset FlightGear interface, use this form [http://www.mathworks.com/contact\\_TS.html](http://www.mathworks.com/contact_TS.html).
- To report bugs or request enhancements to FlightGear itself, visit [www.flightgear.org](http://www.flightgear.org) and use the contact page.

### Obtain FlightGear

You can obtain FlightGear from [www.flightgear.org](http://www.flightgear.org) in the download area or by ordering CDs from FlightGear. The download area contains extensive documentation for installation and configuration. Because FlightGear is an open source project, source downloads are also available for customization and porting to custom environments.

## Configure Your Computer for FlightGear

You must have a high performance graphics card with stable drivers to use FlightGear. For more information, see the FlightGear CD distribution or the hardware requirements and documentation areas of the FlightGear Web site, [www.flightgear.org](http://www.flightgear.org).

MathWorks tests of FlightGear performance and stability indicate significant sensitivity to computer video cards, driver versions, and driver settings. You need OpenGL<sup>®</sup> support with hardware acceleration activated. The OpenGL settings are particularly important. Without proper setup, performance can drop from about a 30 frames-per-second (fps) update rate to less than 1 fps. If your system allows you to update OpenGL settings, modify them to improve performance.

## Graphics Recommendations for Windows

MathWorks recommends the following for Windows users:

- Choose a graphics card with good OpenGL performance.
- Always use the latest tested and stable driver release for your video card. Test the driver thoroughly on a few computers before deploying to others.

For Microsoft Windows XP systems running on x86 (32-bit) or AMD-64/EM64T chip architectures, the graphics card operates in the unprotected kernel space known as Ring Zero. This means that glitches in the driver can cause Windows to lock or crash. Before buying a large number of computers for 3-D applications, test, with your vendor, one or two computers to find a combination of hardware, operating system, drivers, and settings that are stable for your applications.

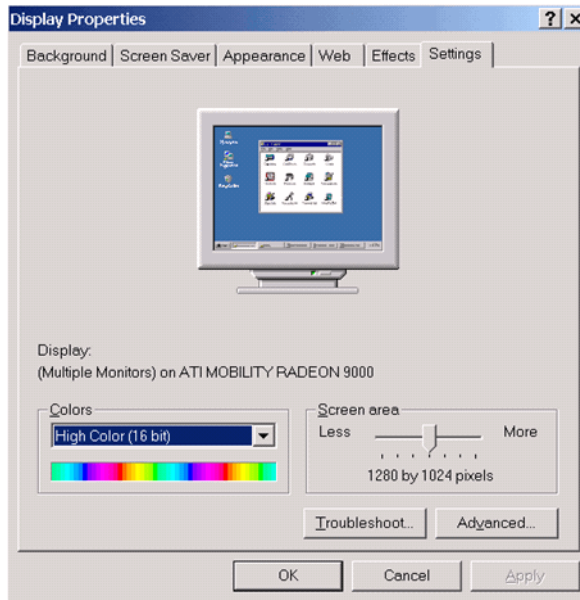
## Set Up OpenGL Graphics on Windows

For complete information on OpenGL settings, refer to the documentation at the OpenGL Web site: [www.opengl.org](http://www.opengl.org).

If your system allows you to modify OpenGL, perform steps like the following to optimize your video card settings. Your driver's panes might look different.

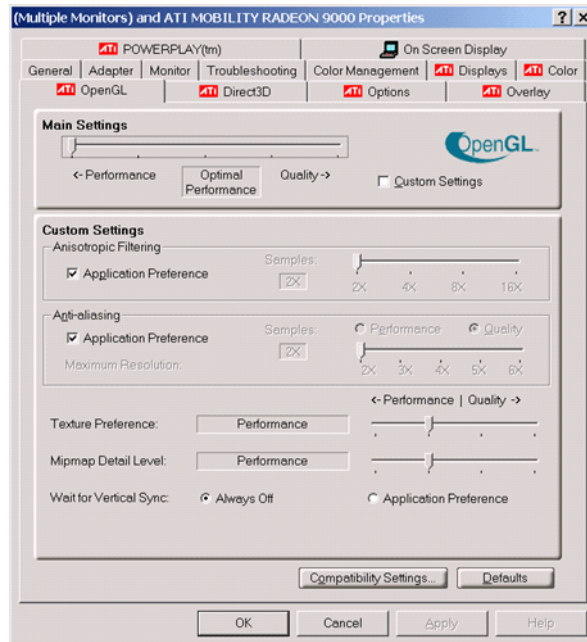
- 1 Ensure that you have activated the OpenGL hardware acceleration on your video card. On Windows, access this configuration through **Start >**

**Settings > Control Panel > Display**, which opens the following dialog box. Select the **Settings** tab.



**2** Click the **Advanced** button in the lower right of the dialog box, which brings up the graphics card's custom configuration dialog box, and go to the **OpenGL** tab. For an ATI Mobility Radeon 9000 video card, the **OpenGL** pane looks like this:





- 3** For best performance, move the **Main Settings** slider near the top of the dialog box to the **Performance** end of the slider.
- 4** If stability is a problem, try other screen resolutions, other color depths in the **Displays** pane, and other OpenGL acceleration modes.

Many cards perform much better at 16 bits-per-pixel color depth (also known as 65536 color mode, 16-bit color). For example, on an ATI Mobility Radeon 9000 running a given model, 30 fps are achieved in 16-bit color mode, while 2 fps are achieved in 32-bit color mode.

### Setup on Linux, Macintosh, and Other Platforms

FlightGear distributions are available for Linux<sup>®</sup>, Macintosh, and other UNIX platforms from the FlightGear Web site, [www.flightgear.org](http://www.flightgear.org). Installation on these platforms, like Windows, requires careful configuration of graphics cards and drivers. Consult the documentation and hardware requirements sections at the FlightGear Web site.

### **Use MATLAB Graphics Controls to Configure Your OpenGL Settings**

You can also control your OpenGL rendering from the MATLAB command line with the MATLAB Graphics `opengl` command. Consult the `opengl` command reference for more information.

### **Install and Start FlightGear**

The extensive FlightGear documentation guides you through the installation in detail. Consult the following:

- Documentation section of the FlightGear Web site for installation instructions: <http://mapserver.flightgear.org/getstart/> located at [www.flightgear.org](http://www.flightgear.org).
- MATLAB system requirements.

Keep the following points in mind:

- Configure your computer's graphics card before you install FlightGear. See the preceding section, "Configure Your Computer for FlightGear" on page 2-23.
- Shut down all running applications (including the MATLAB interface) before installing FlightGear.
- MathWorks tests indicate that the operational stability of FlightGear is especially sensitive during startup. It is best to not move, resize, mouse over, overlap, or cover up the FlightGear window until the initial simulation scene appears after the startup splash screen fades out.

Aerospace Blockset supports FlightGear on a number of platforms (<http://www.mathworks.com/products/aeroblks/requirements.html>). The following table lists the properties you should be aware of before you start to use FlightGear.

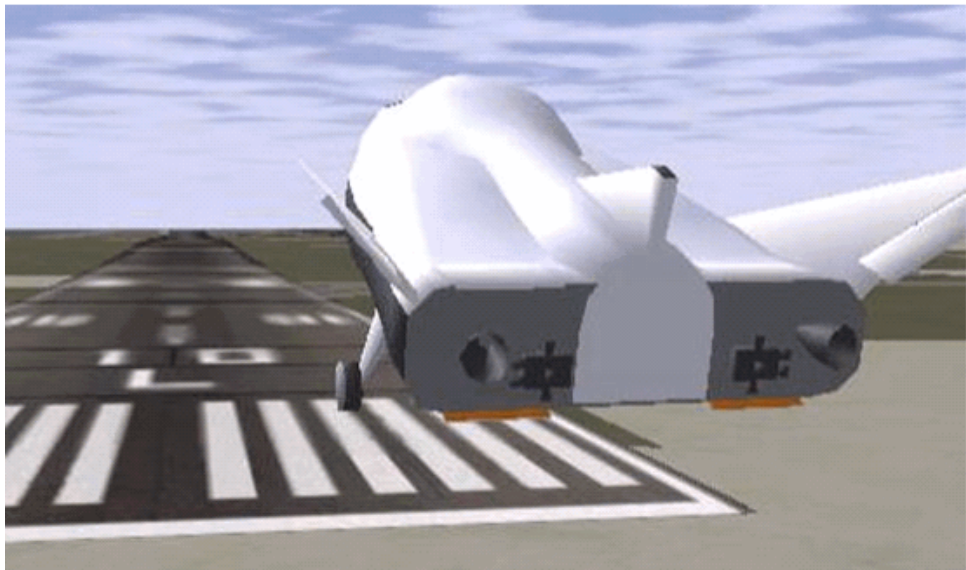
<b>FlightGear Property</b>	<b>Folder Description</b>	<b>Platforms</b>	<b>Typical Location</b>
FlightGearBase-Directory	FlightGear installation folder.	Windows 64-bit	C:\Program Files\FlightGear (default)
		Windows 32-bit	C:\Program Files (x86)\-FlightGear (default)
		Solaris™ or Linux	Folder into which you installed FlightGear
		Mac	/Applications (directory to which you dragged the FlightGear icon)
GeometryModelName	Model geometry folder	Windows 64-bit	C:\Program Files\FlightGear\data\Aircraft\HL20 (default)
		Windows 32-bit	C:\Program Files (x86)\-FlightGear\data\Aircraft\HL20 (default)
		Solaris or Linux	\$FlightGearBaseDirectory/-data/Aircraft/HL20
		Mac	\$FlightGearBaseDirectory/-FlightGear.app/Contents/Resources/data/Aircraft/HL20

## Work with the Flight Simulator Interface

In this section...
“Introduction” on page 2-28
“About Aircraft Geometry Models” on page 2-29
“Work with Aircraft Geometry Models” on page 2-31
“Run FlightGear with the Simulink Models” on page 2-34
“Run the NASA HL-20 Example with FlightGear” on page 2-42
“Send and Receive Data” on page 2-45

### Introduction

Use this section to learn how to use the FlightGear flight simulator and the Aerospace Blockset software to visualize your Simulink aircraft models. If you have not yet installed FlightGear, see “Flight Simulator Interface” on page 2-22 first.



**Simulink® Driven HL-20 Model in a Landing Flare at KSFC**

## About Aircraft Geometry Models

Before you can visualize your aircraft's dynamics, you need to create or obtain an aircraft model file compatible with FlightGear. This section explains how to do this.

### Aircraft Geometry Editors and Formats

You have a competitive choice of over twelve 3-D geometry file formats supported by FlightGear.

Currently, the most popular 3-D geometry file format is the AC3D format, which has the suffix `*.ac`. AC3D is a low-cost geometry editor available from [www.ac3d.org](http://www.ac3d.org). Another popular 3-D editor for aircraft models is Flight Sim Design Studio, distributed by Abacus Publications at [www.abacuspub.com](http://www.abacuspub.com).

### Aircraft Model Structure and Requirements

Aircraft models are contained in the `FlightGearRoot/data/Aircraft/` folder and subfolders. A complete aircraft model must contain a folder linked through the required aircraft master file named `model-set.xml`.

All other model elements are optional. This is a partial list of the optional elements you can put in an aircraft data folder:

- Vehicle objects and their shapes and colors
- Vehicle objects' surface bitmaps
- Variable geometry descriptions
- Cockpit instrument 3-D models
- Vehicle sounds to tie to events (e.g., engine, gear, wind noise)
- Flight dynamics model
- Simulator views
- Submodels (independently movable items) associated with the vehicle

Model behavior reverts to defaults when these elements are not used. For example,

- Default sound: no vehicle-related sounds are emitted.

- Default instrument panel: no instruments are shown.

Models can contain some, all, or even none of the above elements. If you always run FlightGear from the cockpit view, the aircraft geometry is often secondary to the instrument geometries.

A how-to document for including optional elements is included in the FlightGear documentation at:

<http://www.flightgear.org/Docs/fgfs-model-howto.html>

### **Required Flight Dynamics Model Specification**

The flight dynamics model (FDM) specification is a required element for an aircraft model. To set the Simulink software as the source of the flight dynamics model data stream for a given geometry model, you put this line in `data/Aircraft/model/model-set.xml`:

```
<flight-model>network</flight-model>
```

### **Obtain and Modify Existing Aircraft Models**

You can quickly build models from scratch by referencing instruments, sounds, and other optional elements from existing FlightGear models. Such models provide examples of geometry, dynamics, instruments, views, and sounds. It is simple to copy an aircraft folder to a new name, rename the `model-set.xml` file, modify it for network flight dynamics, and then run FlightGear with the `aircraft` flag set to the name in `model-set.xml`.

Many existing 3-D aircraft geometry models are available for use with FlightGear. Visit the download area of [www.flightgear.org](http://www.flightgear.org) to see some of the aircraft models available. Additional models can be obtained via Web search. Search key words such as “flight gear aircraft model” are a good starting point. Be sure to comply with copyrights when distributing these files.

### **Hardware Requirements for Aircraft Geometry Rendering**

When creating your own geometry files, keep in mind that your graphics card can efficiently render a limited number of surfaces. Some cards can efficiently render fewer than 1000 surfaces with bitmaps and specular reflections at

the nominal rate of 30 frames per second. Other cards can easily render on the order of 10,000 surfaces.

If your performance slows while using a particular geometry, gauge the effect of geometric complexity on graphics performance by varying the number of aircraft model surfaces. An easy way to check this is to replace the full aircraft geometry file with a simple shape, such as a single triangle, then test FlightGear with this simpler geometry. If a geometry file is too complex for smooth display, use a 3-D geometry editor to simplify your model by reducing the number of surfaces in the geometry.

## Work with Aircraft Geometry Models

Once you have obtained, modified, or created an aircraft data file, you need to put it in the correct folder for FlightGear to access it.

### Import Aircraft Models into FlightGear

To install a compatible model into FlightGear, use one of the following procedures. Choose the one appropriate for your platform. This section assumes that you have read “Install and Start FlightGear” on page 2-26.

If your platform is Windows:

- 1 Go to your installed FlightGear folder. Open the data folder, then the Aircraft folder: `\FlightGear\data\Aircraft\`.
- 2 Make a subfolder `model\` here for your aircraft data.
- 3 Put `model-set.xml` in that subfolder, plus any other files needed.

It is common practice to make subdirectories for the vehicle geometry files (`\model\`), instruments (`\instruments\`), and sounds (`\sounds\`).

If your platform is UNIX or Linux:

- 1 Go to your installed FlightGear directory. Open the data directory, then the Aircraft directory: `$FlightGearBaseDirectory/data/Aircraft/`.
- 2 Make a subdirectory `model/` here for your aircraft data.

- 3 Put `model-set.xml` in that subdirectory, plus any other files needed.

It is common practice to make subdirectories for the vehicle geometry files (`/model/`), instruments (`/instruments/`), and sounds (`/sounds/`).

If your platform is Mac:

- 1 Open a terminal.
- 2 Go to your installed FlightGear folder. Open the data folder, then the Aircraft folder:

```
$FlightGearBaseDirectory/FlightGear.app/Contents/Resources/data/Aircraft/
```

- 3 Make a subfolder `model/` here for your aircraft data.
- 4 Put `model-set.xml` in that subfolder, plus any other files needed.

It is common practice to make subdirectories for the vehicle geometry files (`/model/`), instruments (`/instruments/`), and sounds (`/sounds/`).

### **Example: Animate Vehicle Geometries**

This example illustrates how to prepare hinge line definitions for animated elements such as vehicle control surfaces and landing gear. To enable animation, each element must be a named entity in a geometry file. The resulting code forms part of the HL20 lifting body model presented in “Run the NASA HL-20 Example with FlightGear” on page 2-42.

- 1 The standard body coordinates used in FlightGear geometry models form a right-handed system, rotated from the standard body coordinate system in *Y* by -180 degrees:
  - *X* = positive toward the back of the vehicle
  - *Y* = positive toward the right of the vehicle
  - *Z* = positive is up, e.g., wheels typically have the lowest *Z* values.

See “About Aerospace Coordinate Systems” on page 2-12 for more details.



- 2** Find two points that lie on the desired named-object hinge line in body coordinates and write them down as *XYZ* triplets or put them into a MATLAB calculation like this:

```
a = [2.98, 1.89, 0.53];
b = [3.54, 2.75, 1.46];
```

- 3** Calculate the difference between the points:

```
pdiff = b - a
pdiff =
    0.5600    0.8600    0.9300
```

- 4** The hinge point is either of the points in step 2 (or the midpoint as shown here):

```
mid = a + pdiff/2
mid =
    3.2600    2.3200    0.9950
```

- 5** Put the hinge point into the animation scope in *model-set.xml*:

```
<center>
  <x-m>3.26</x-m>
  <y-m>2.32</y-m>
  <z-m>1.00</z-m>
</center>
```

- 6** Use the difference from step 3 to define the relative motion vector in the animation axis:

```
<axis>
  <x>0.56</x>
  <y>0.86</y>
  <z>0.93</z>
</axis>
```

- 7** Put these steps together to obtain the complete hinge line animation used in the HL20 example model:

```
<animation>
```

```
<type>rotate</type>
<object-name>RightAileron</object-name>
<property>/surface-positions/right-aileron-pos-norm</property>
<factor>30</factor>
<offset-deg>0</offset-deg>
<center>
  <x-m>3.26</x-m>
  <y-m>2.32</y-m>
  <z-m>1.00</z-m>
</center>
<axis>
  <x>0.56</x>
  <y>0.86</y>
  <z>0.93</z>
</axis>
</animation>
```

## Run FlightGear with the Simulink Models

To run a Simulink model of your aircraft and simultaneously animate it in FlightGear with an aircraft data file *model-set.xml*, you need to configure the aircraft data file and modify your Simulink model with some new blocks.

These are the main steps to connecting and using FlightGear with the Simulink software:

- “Set the Flight Dynamics Model to Network in the Aircraft Data File” on page 2-35 explains how to create the network connection you need.
- “Obtain the Destination IP Address” on page 2-35 starts by determining the IP address of the computer running FlightGear.
- “Send Simulink Data to FlightGear” on page 2-45 shows how to add and connect interface and pace blocks to your Simulink model.
- “Create a FlightGear Run Script” on page 2-36 shows how to write a FlightGear run script compatible with your Simulink model.
- “Start FlightGear” on page 2-39 guides you through the final steps to making the Simulink software work with FlightGear.
- “Improve Performance” on page 2-40 helps you speed your model up.

- “Run FlightGear and Simulink Software on Different Computers” on page 2-41 explains how to connect a simulation from the Simulink software running on one computer to FlightGear running on another computer.

## Set the Flight Dynamics Model to Network in the Aircraft Data File

Be sure to:

- Remove any pre-existing flight dynamics model (FDM) data from the aircraft data file.
- Indicate in the aircraft data file that its FDM is streaming from the network by adding this line:

```
<flight-model>network</flight-model>
```

## Obtain the Destination IP Address

You need the destination IP address for your Simulink model to stream its flight data to FlightGear.

- If you know your computer’s name, enter at the MATLAB command line:

```
java.net.InetAddress.getByName( 'www.mathworks.com' )
```

- If you are running FlightGear and the Simulink software on the same computer, get your computer’s name by entering at the MATLAB command line:

```
java.net.InetAddress.getLocalHost
```

- If you are working in Windows, get your computer’s IP address by entering at the DOS prompt:

```
ipconfig /all
```

Examine the IP address entry in the resulting output. There is one entry per Ethernet device.

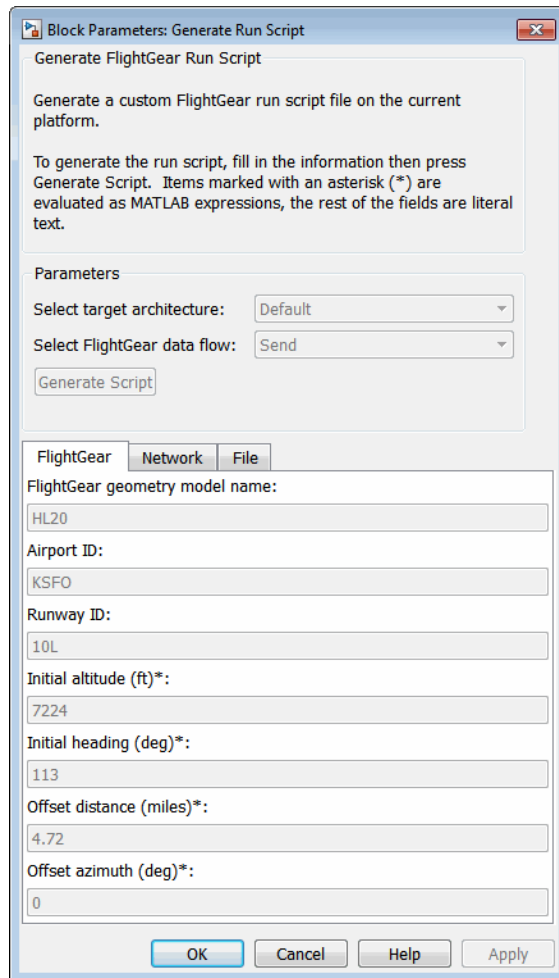
### **Create a FlightGear Run Script**

To start FlightGear with the desired initial conditions (location, date, time, weather, operating modes), it is best to create a run script by “Use the Generate Run Script Block” on page 2-36 or “Use the Interface Provided with FlightGear” on page 2-39.

If you make separate run scripts for each model you intend to link to FlightGear and place them in separate directories, run the appropriate script from the MATLAB interface just before starting your Simulink model.

**Use the Generate Run Script Block.** The easiest way to create a run script is by using the Generate Run Script block. Use the following procedure:

- 1** Open the Flight Simulator Interfaces sublibrary.
- 2** Create a new Simulink model or open an existing model.
- 3** Drag a Generate Run Script block into the Simulink diagram.
- 4** Double-click the Generate Run Script block. Its dialog opens. Observe the three panes, **FlightGear**, **Network**, and **File**.



- 5 In the **File > Output file name** field, type the name of the output file. This name should be the name of the command you want to use to start FlightGear with these initial parameters. Use the appropriate file extension:

<b>Platform</b>	<b>Extension</b>
Windows	.bat
Linux and Mac OS	.sh

For example, if your file name is `runfg.bat`, use the `runfg` command to execute the run script and start FlightGear.

- 6** In the **File > FlightGear base directory** field, specify the name of your FlightGear installation folder.
- 7** In the **File > FlightGear geometry model name** field, specify the name of the subfolder, in the `FlightGear/data/Aircraft` folder, containing the desired model geometry.
- 8** Specify the initial conditions as needed.
- 9** Click the **Generate Script** button at the top of the **Parameters** area.

The Aerospace Blockset software generates the run script, and saves it in your MATLAB working folder under the file name that you specified in the **File > Output file name** field.

- 10** Repeat steps 5 through 9 to generate other run scripts, if needed.
- 11** Click **OK** to close the dialog box. You do not need to save the Generate Run Script block with the Simulink model.

The Generate Run Script block saves the run script as a text file in your working folder. This is an example of the contents of a run script file:

```
>> cd D:\Applications\FlightGear-2.8
>> SET FG_ROOT=D:\Applications\FlightGear-2.8\data
>> cd \bin\Win32\
>> fgfs --aircraft=HL20 --fdm=network,localhost,5501,5502,5503
--fog-fastest --disable-clouds --start-date-lat=2004:06:01:09:00:00
--disable-sound --in-air --enable-freeze --airport=KSFO --runway=10L
--altitude=7224 --heading=113 --offset-distance=4.72 --offset-azimuth=0
```

**Use the Interface Provided with FlightGear.** The FlightGear launcher GUI (part of FlightGear, not the Aerospace Blockset product) lets you build simple and advanced options into a visible FlightGear run command.

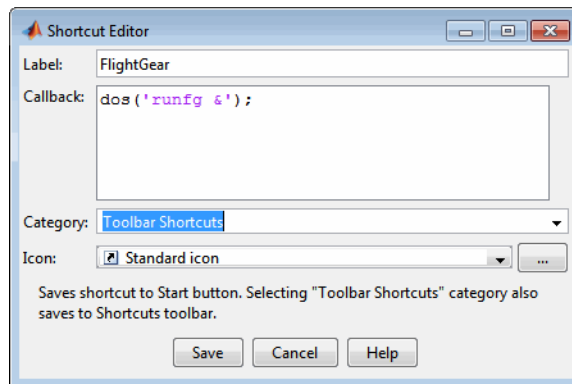
## Start FlightGear

If your computer has enough computational power to run both the Simulink software and FlightGear at the same time, a simple way to start FlightGear on a Windows system is to create a MATLAB desktop button containing the following command to execute a run script like the one created above:

```
system('runfg &')
```

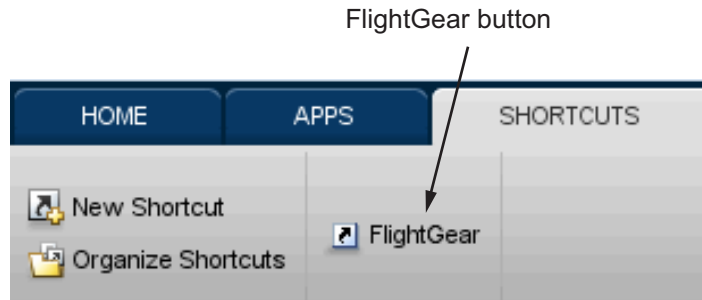
To create a desktop button:

- 1** In the MATLAB Command Window, select the **Shortcuts** and click **New Shortcut**. The **Shortcut Editor** dialog opens.
- 2** Set the **Label**, **Callback**, **Category**, and **Icon** fields as shown in the following figure.



- 3** Click **Save**.

The **FlightGear** button appears in your MATLAB desktop. If you click it, the output file, for example `runfg.bat`, runs in the current folder.



Once you have completed the setup, start FlightGear and run your model:

- 1 Make sure your model is in a writable folder. Open the model, and update the diagram. This step ensures that any referenced block code is compiled and that the block diagram is compiled before running. Once you start FlightGear, it uses all available processor power while it is running.
- 2 Click the **FlightGear** button or run the FlightGear run script manually.
- 3 When FlightGear starts, it displays the initial view at the initial coordinates specified in the run script. If you are running the Simulink software and FlightGear on different computers, arrange to view the two displays at the same time.
- 4 Now begin the simulation and view the animation in FlightGear.

### Improve Performance

If your Simulink model is complex and cannot run at the aggregate rate needed for the visualization, you might need to

- Use the Accelerator mode in Simulink (“Perform Acceleration”).
- Free up processor power by running the Simulink model on one computer and FlightGear on another computer. Use the **Destination IP Address** parameter of the Send net\_fdm Packet to FlightGear block to specify the network address of the computer where FlightGear is running.
- Simulate the Simulink model first, then save the resulting translations (*x*-axis, *y*-axis, *z*-axis) and positions (latitude, longitude, altitude), and use



the FlightGear Animation object in Aerospace Toolbox to visualize this data.

## Run FlightGear and Simulink Software on Different Computers

It is possible to simulate an aerospace system in the Simulink environment on one computer (the source) and use its simulation output to animate FlightGear on another computer (the target). The steps are similar to those already explained, with certain modifications.

- 1** Obtain the IP address of the computer running FlightGear. See “Obtain the Destination IP Address” on page 2-35.
- 2** Enter this target computer’s IP address in the Send net\_fdm Packet to FlightGear block. See “Send Simulink Data to FlightGear” on page 2-45.
- 3** Update the Generate Run Script block in your model with the target computer’s FlightGear base folder. Regenerate the run script to reflect the target computer’s separate identity.

See “Create a FlightGear Run Script” on page 2-36.

- 4** Copy the generated run script to the target computer. Start FlightGear there. See “Start FlightGear” on page 2-39.
- 5** If you want to also receive data from FlightGear, use the Receive net\_ctrl Packet from FlightGear block. Enter the IP address of the computer running FlightGear in the **Origin IP address** parameter.
- 6** Update the run script for the receive data. Use the Generate Run Script block to regenerate the run script.
- 7** Start your Simulink model on the source computer. FlightGear running on the target displays the simulation motion.

## Install Additional FlightGear Scenery

When you install the FlightGear software, the installation provides a basic level of scenery files. The FlightGear documentation guides you through installing scenery as part the general FlightGear installation.

If you need to install more FlightGear scenery files, see the instructions at <http://www.flightgear.org>. The instructions describe how to install the additional scenery in a default location. MathWorks recommends that you follow those instructions.

If you must install additional scenery in a nonstandard location, try setting the `FG_SCENERY` environment variable in the script output from the Generate Run Script block. See the documentation at <http://www.flightgear.org> for a description of the `FG_SCENERY` variable.

---

**Note** Each time that you click the **Generate Script** button, the Generate Run Script block creates a new script and overwrites any edits that you have added.

---

### Run the NASA HL-20 Example with FlightGear

The Aerospace Blockset software contains an example model of the NASA HL-20 lifting body that uses the FlightGear interface.

Before attempting to simulate this model, you must have FlightGear installed and configured. See “Flight Simulator Interface” on page 2-22. Also, before attempting to simulate this model, read “Install and Start FlightGear” on page 2-26. Before you start, perform the following actions, depending on your platform.

#### Windows

Copy the HL20 folder from `matlabroot\toolbox\aeoblks\aedemos\` folder to `FlightGear\data\Aircraft\` folder. This folder contains the preconfigured geometries for the HL-20 simulation and `HL20-set.xml`. The file `matlabroot\toolbox\aeoblks\aedemos\HL20\models\HL20.xml` defines the geometry.

For more information, see “Import Aircraft Models into FlightGear” on page 2-31.

#### UNIX/Linux

Copy the HL20 directory from `matlabroot/toolbox/aeoblks/aedemos/` directory to

*\$FlightGearBaseDirectory/data/Aircraft/* directory.

This directory contains the preconfigured geometries for the HL-20 simulation and *HL20-set.xml*. The file *matlabroot/toolbox/aeroblks/aerodemos/HL20/models/HL20.xml* defines the geometry.

For more about this step, see “Import Aircraft Models into FlightGear” on page 2-31.

#### Mac

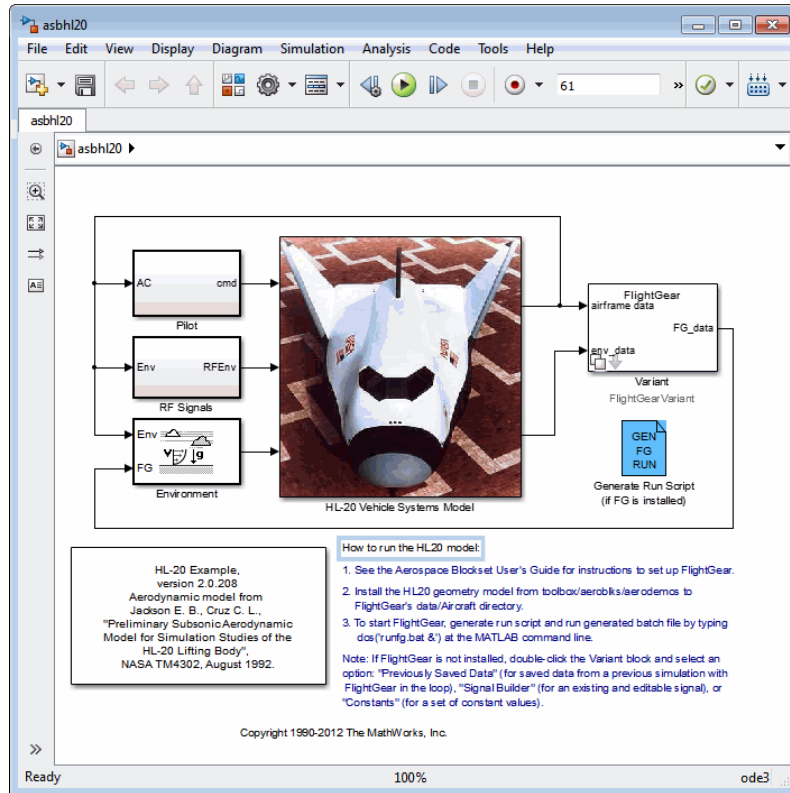
Copy the HL20 folder from

*matlabroot/toolbox/aeroblks/aerodemos/* folder to

*\$FlightGearBaseDirectory/FlightGear.app/Contents/Resources/-data/Aircraft/* folder. This folder contains the preconfigured geometries for the HL-20 simulation and *HL20-set.xml*. The file *matlabroot\toolbox\ aeroblks\ aerodemos\ HL20\ models\ HL20.xml* defines the geometry.

For more about this step, see “Import Aircraft Models into FlightGear” on page 2-31.

- 1 Start the MATLAB interface. Open the example either by entering *asbh120* in the MATLAB Command Window or by finding the example entry (NASA HL-20 with FlightGear Interface) in the Aerospace Blockset Examples page. The model opens.



**2** If this is your first time running FlightGear for this model, double-click the Generate Run Script block to create a run script. Make sure to specify your FlightGear installation folder in the **File > FlightGear base directory** field. For more information, see “Create a FlightGear Run Script” on page 2-36.

**3** Execute the script you just created manually by entering the following at the MATLAB command line:

```
system('runfg &')
```

If you created a **FlightGear** desktop button, you can click it instead to start the run script and start FlightGear. For more information, see “Start FlightGear” on page 2-39.

**4** Now start the simulation and view the animation in FlightGear.

---

**Note** With the FlightGear window in focus, press the **V** key to alternate between the different aircraft views: cockpit view, helicopter view, chase view, and so on.

---

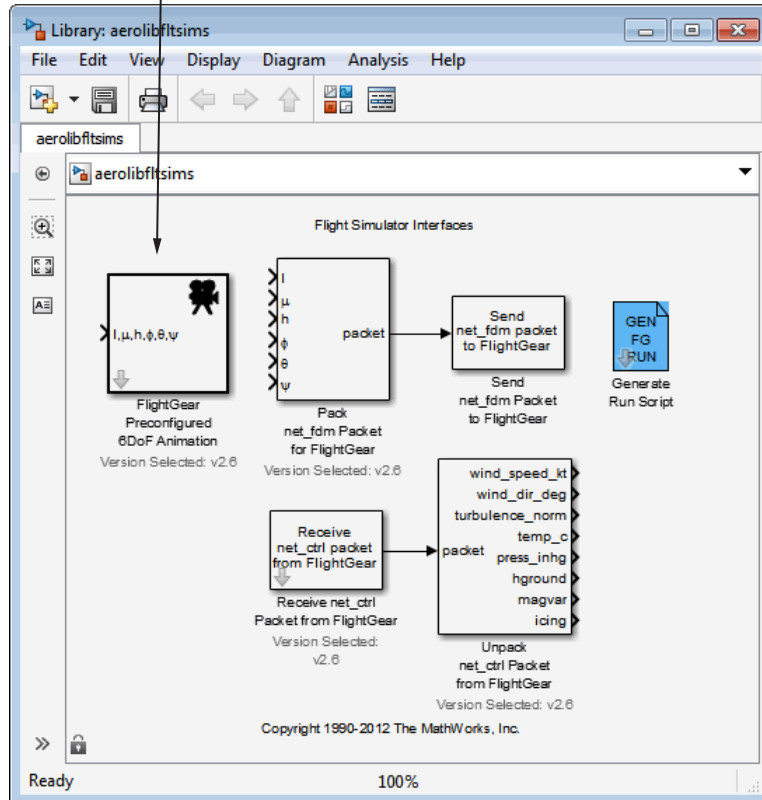
## **Send and Receive Data**

You can send and receive data between a Simulink model and a running FlightGear Flight Simulator.

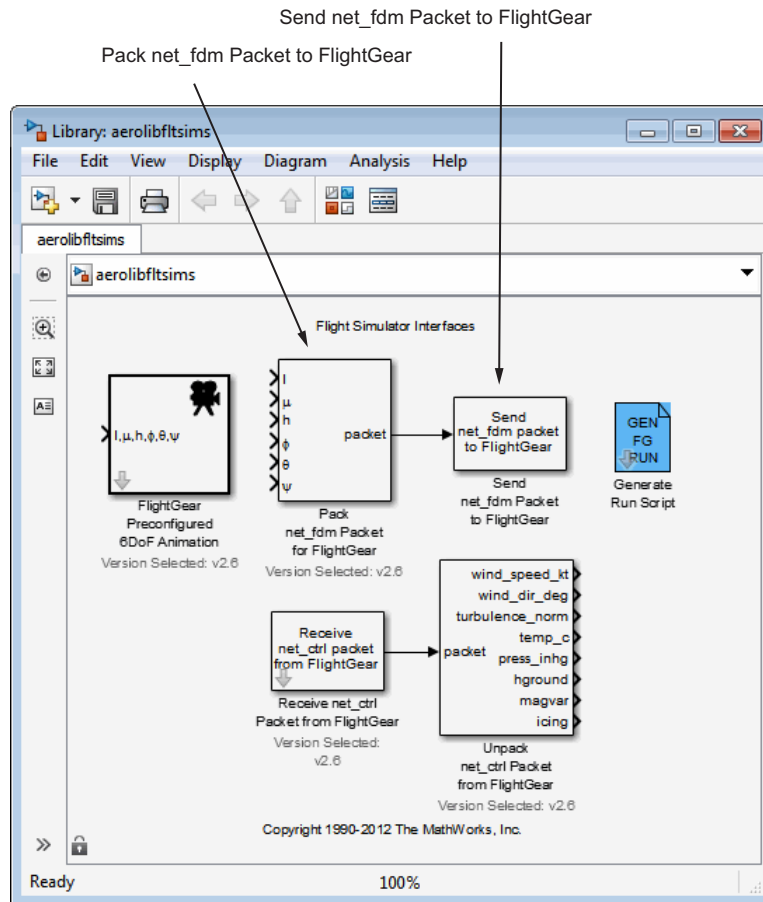
### **Send Simulink Data to FlightGear**

The easiest way to connect your model to FlightGear with the blockset is to use the FlightGear Preconfigured 6DoF Animation block:

FlightGear Preconfigured 6DoF Animation Block



The FlightGear Preconfigured 6DoF Animation block is a subsystem containing the Pack net\_fdm Packet for FlightGear and Send net\_fdm Packet to FlightGear blocks:



These blocks transmit data from a model to a FlightGear session. The blocks are separate for maximum flexibility and compatibility.

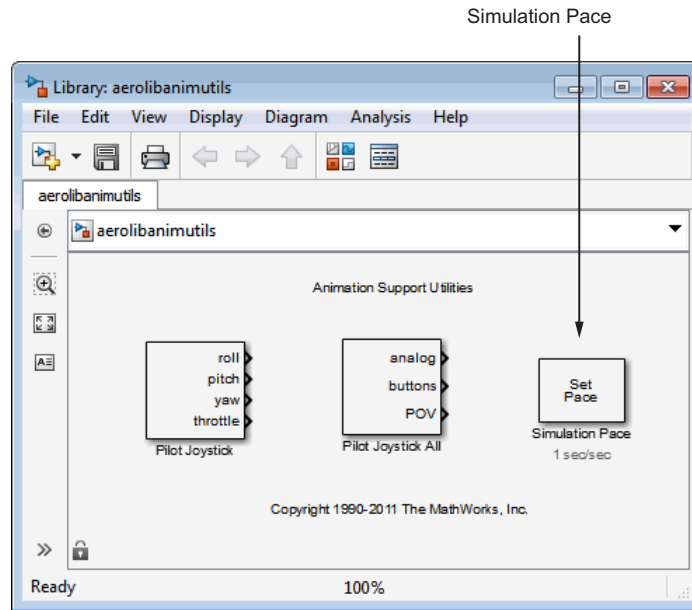
- The Pack net\_fdm Packet for FlightGear block formats a binary structure compatible with FlightGear from model inputs. In the default configuration, the block displays only the 6DoF ports, but you can configure the full FlightGear interface supporting more than 50 distinct signals from the block dialog box:

```
l
u
phi
theta
psi
alpha
beta
gamma
delta/dt
delta/dt
du/dt
v
cmfb_rate
v_north
v_east
v_down
v_wind_body_north
v_wind_body_east
v_wind_body_down
A_X_pilot
A_Y_pilot
stall_warning
slip_deg
elevator
elevator_trim_tab
left_flap
right_flap
left_aileron packet
right_aileron
rudder
nose_wheel
speedbrake
spoilers
num_engines
eng_state
rpm
fuel_flow
fuel_px
egt
cht
mp_osi
tit
oil_temp
oil_px
num_tanks
fuel_quantity
num_wheels
wow
gear_pos
gear_steer
gear_compression
agl
cur_time
warp
visibility
```

Packet  
net\_fdm Packet  
for FlightGear  
Version Selected: v2.6

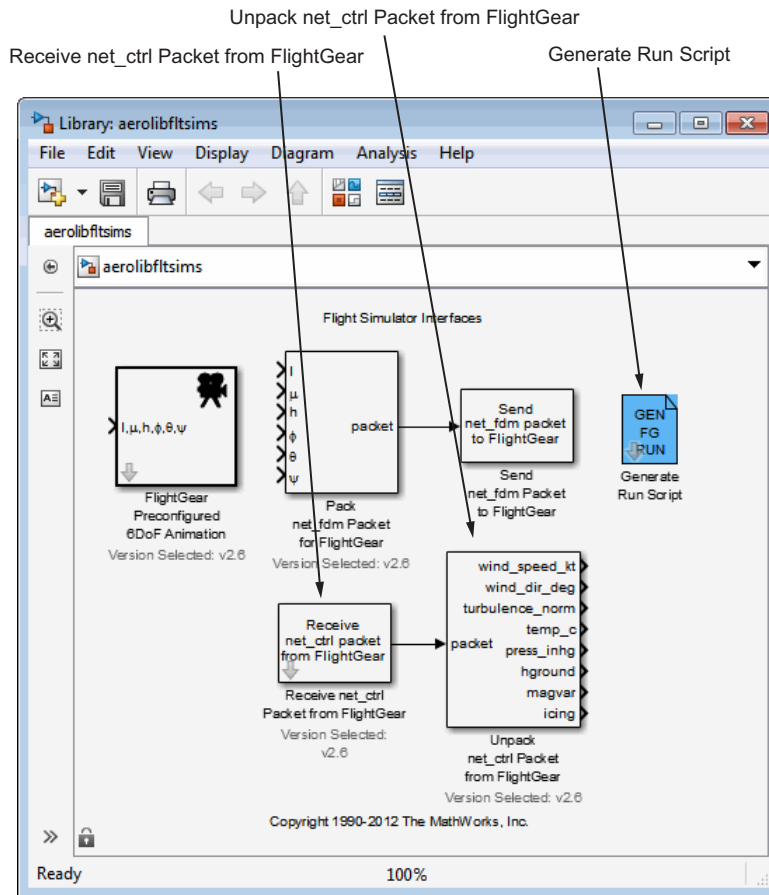
- The Send net\_fdm Packet to FlightGear block transmits this packet via UDP to the specified IP address and port where a FlightGear session awaits an incoming datastream. Use the IP address you found in “Obtain the Destination IP Address” on page 2-35.
- The Simulation Pace block slows the simulation so that its aggregate run rate is 1 second of simulation time per second of clock time. You can also use it to specify other ratios of simulation time to clock time.





## Send FlightGear Data to Model

To increase the accuracy of your model simulation, you might want to send FlightGear environment variables to the Simulink model. Use the following blocks:



- **Receive net\_ctrl Packet from FlightGear** — Receives a network control and environment data packet `net_ctrl` from either the simulation of a Simulink model in the FlightGear simulator, or from a FlightGear session.
- **Unpack net\_ctrl Packet from FlightGear** — Unpacks `net_ctrl` variable packets received from FlightGear and makes them available for the Simulink environment.
- **Generate Run Script** — Generates a customized FlightGear run script on the current platform.

For an example of how to use these blocks to send data to a Simulink model, see [NASA HL-20 with FlightGear Interface](#).

These blocks use UDP to transfer data from FlightGear to the Simulink environment. Note the following:

- When a host and target are Windows or Linux platforms, you can use any combination of Windows or Linux platforms for the host and target.
- When a host or target is a Mac platform, use only Mac platforms for both the host and target.

### **Macintosh Platform and FlightGear 2.6**

On a Macintosh system with FlightGear 2.6, you might see unexpected results (for example, very large or very small data values) if your model uses the following blocks:

- Receive net\_ctrl Packet from FlightGear
- Unpack net\_ctrl Packet from FlightGear

To work around this issue:

- 1** In the model, change the **FlightGear version** parameter to **v2.4** for both blocks.
- 2** Save and rerun the model.

The results should now be as expected.



# Case Studies

---

- “Ideal Airspeed Correction” on page 3-2
- “1903 Wright Flyer” on page 3-10
- “NASA HL-20 Lifting Body Airframe” on page 3-20

## Ideal Airspeed Correction

In this section...
“Introduction” on page 3-2
“Airspeed Correction Models” on page 3-2
“Measure Airspeed” on page 3-4
“Model Airspeed Correction” on page 3-5
“Simulate Airspeed Correction” on page 3-7

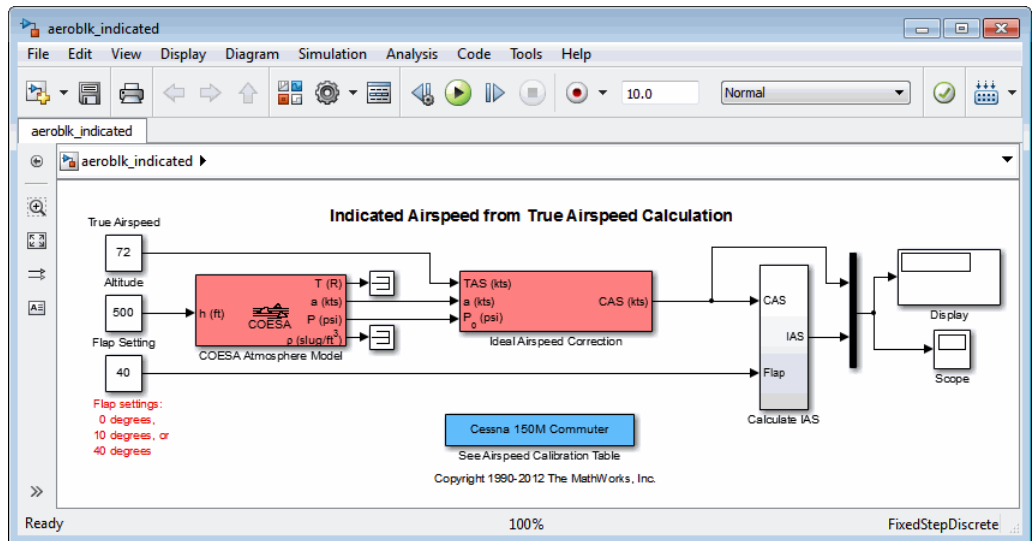
### Introduction

This case study simulates indicated and true airspeed. It constitutes a fragment of a complete aerodynamics problem, including only measurement and calibration.

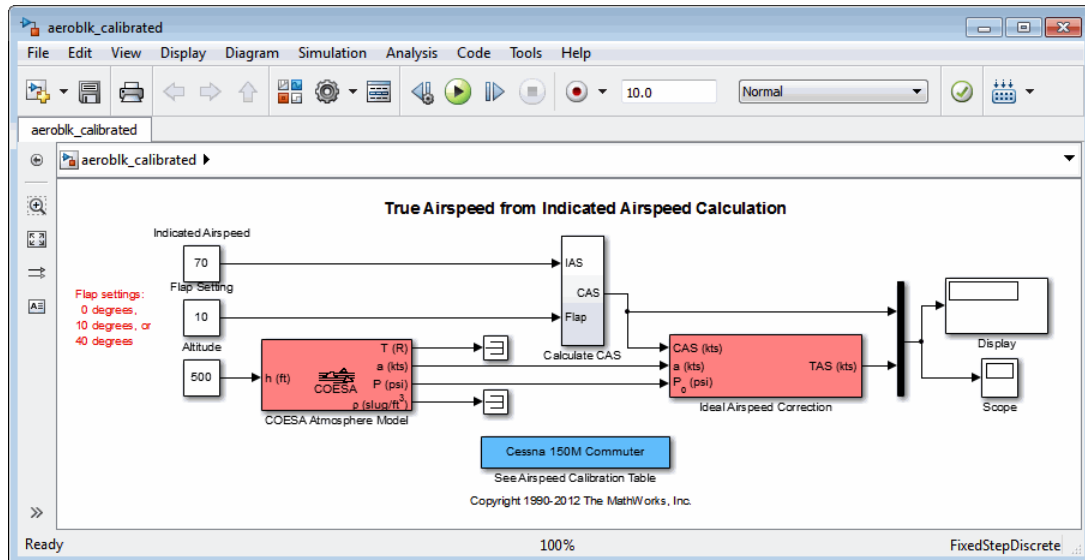
### Airspeed Correction Models

To view the airspeed correction models, enter the following at the MATLAB command line:

```
aeroblk_indicated  
aeroblk_calibrated
```



**aeroblk\_indicated Model**



**aeroblk\_calibrated Model**

## Measure Airspeed

To measure airspeed, most light aircraft designs implement pitot-static airspeed indicators based on Bernoulli's principle. Pitot-static airspeed indicators measure airspeed by an expandable capsule that expands and contracts with increasing and decreasing dynamic pressure. This is known as *calibrated airspeed* (CAS). It is what a pilot sees in the cockpit of an aircraft.

To compensate for measurement errors, it helps to distinguish three types of airspeed. These types are explained more completely in the following.

Airspeed Type	Description
Calibrated	Indicated airspeed corrected for calibration error
Equivalent	Calibrated airspeed corrected for compressibility error
True	Equivalent airspeed corrected for density error

### Calibration Error

An airspeed sensor features a static vent to maintain its internal pressure equal to atmospheric pressure. Position and placement of the static vent with respect to the angle of attack and velocity of the aircraft determines the pressure inside the airspeed sensor and therefore the calibration error. Thus, a calibration error is specific to an aircraft's design.

An airspeed calibration table, which is usually included in the pilot operating handbook or other aircraft documentation, helps pilots convert the indicated airspeed to the calibrated airspeed.

### Compressibility Error

The density of air is not constant, and the compressibility of air increases with altitude and airspeed, or when contained in a restricted volume. A pitot-static airspeed sensor contains a restricted volume of air. At high altitudes and high airspeeds, calibrated airspeed is always higher than equivalent airspeed. Equivalent airspeed can be derived by adjusting the calibrated airspeed for compressibility error.



## Density Error

At high altitudes, airspeed indicators read lower than true airspeed because the air density is lower. True airspeed represents the compensation of equivalent airspeed for the density error, the difference in air density at altitude from the air density at sea level, in a standard atmosphere.

## Model Airspeed Correction

The `aeroblk_indicated` and `aeroblk_calibrated` models show how to take true airspeed and correct it to indicated airspeed for instrument display in a Cessna 150M Commuter light aircraft. The `aeroblk_indicated` model implements a conversion to indicated airspeed. The `aeroblk_calibrated` model implements a conversion to true airspeed.

Each model consists of two main components:

- “COESA Atmosphere Model Block” on page 3-5 calculates the change in atmospheric conditions with changing altitude.
- “Ideal Airspeed Correction Block” on page 3-5 transforms true airspeed to calibrated airspeed and vice versa.

## COESA Atmosphere Model Block

The COESA Atmosphere Model block is a mathematical representation of the U.S. 1976 COESA (Committee on Extension to the Standard Atmosphere) standard lower atmospheric values for absolute temperature, pressure, density, and speed of sound for input geopotential altitude. Below 32,000 meters (104,987 feet), the U.S. Standard Atmosphere is identical with the Standard Atmosphere of the ICAO (International Civil Aviation Organization).

The `aeroblk_indicated` and `aeroblk_calibrated` models use the COESA Atmosphere Model block to supply the speed of sound and air pressure inputs for the Ideal Airspeed Correction block in each model.

## Ideal Airspeed Correction Block

The Ideal Airspeed Correction block compensates for airspeed measurement errors to convert airspeed from one type to another type. The following table contains the Ideal Airspeed Correction block’s inputs and outputs.

<b>Airspeed Input</b>	<b>Airspeed Output</b>
True Airspeed	Equivalent airspeed Calibrated airspeed
Equivalent Airspeed	True airspeed Calibrated airspeed
Calibrated Airspeed	True airspeed Equivalent airspeed

In the `aeroblk_indicated` model, the Ideal Airspeed Correction block transforms true to calibrated airspeed. In the `aeroblk_calibrated` model, the Ideal Airspeed Correction block transforms calibrated to true airspeed.

The following sections explain how the Ideal Airspeed Correction block mathematically represents airspeed transformations:

- “True Airspeed Implementation” on page 3-6
- “Calibrated Airspeed Implementation” on page 3-7
- “Equivalent Airspeed Implementation” on page 3-7

**True Airspeed Implementation.** True airspeed (TAS) is implemented as an input and as a function of equivalent airspeed (EAS), expressible as

$$TAS = \frac{EAS \times a}{a_0 \sqrt{\delta}}$$

where

- $a$  Speed of sound at altitude in m/s
- $\delta$  Relative pressure ratio at altitude
- $a_0$  Speed of sound at mean sea level in m/s

**Calibrated Airspeed Implementation.** Calibrated airspeed (CAS), derived using the compressible form of Bernoulli's equation and assuming isentropic conditions, can be expressed as

$$CAS = \sqrt{\frac{2\gamma P_0}{(\gamma - 1)\rho_0} \left[ \left( \frac{q}{P_0} + 1 \right)^{(\gamma-1)/\gamma} - 1 \right]}$$

where

$\rho_0$	Air density at mean sea level in kg/m <sup>3</sup>
$P_0$	Static pressure at mean sea level in N/m <sup>2</sup>
$\gamma$	Ratio of specific heats
$q$	Dynamic pressure at mean sea level in N/m <sup>2</sup>

**Equivalent Airspeed Implementation.** Equivalent airspeed (EAS) is the same as CAS, except static pressure at sea level is replaced by static pressure at altitude.

$$EAS = \sqrt{\frac{2\gamma P}{(\gamma - 1)\rho_0} \left[ \left( \frac{q}{P} + 1 \right)^{(\gamma-1)/\gamma} - 1 \right]}$$

The symbols are defined as follows:

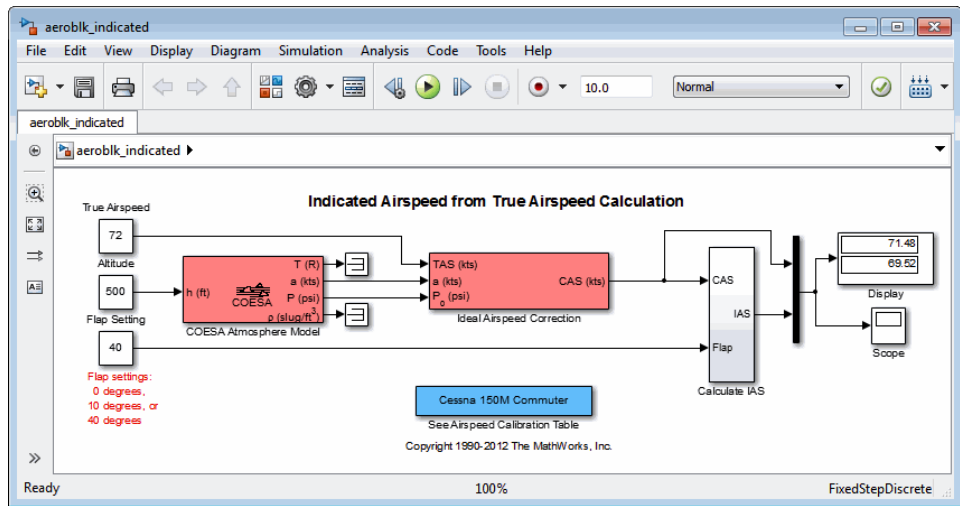
$\rho_0$	Air density at mean sea level in kg/m <sup>3</sup>
$P$	Static pressure at altitude in N/m <sup>2</sup>
$\gamma$	Ratio of specific heats
$q$	Dynamic pressure at mean sea level in N/m <sup>2</sup>

## Simulate Airspeed Correction

In the aeroblk\_indicated model, the aircraft is defined to be traveling at a constant speed of 72 knots (true airspeed) and altitude of 500 feet. The flaps are set to 40 degrees. The COESA Atmosphere Model block takes the

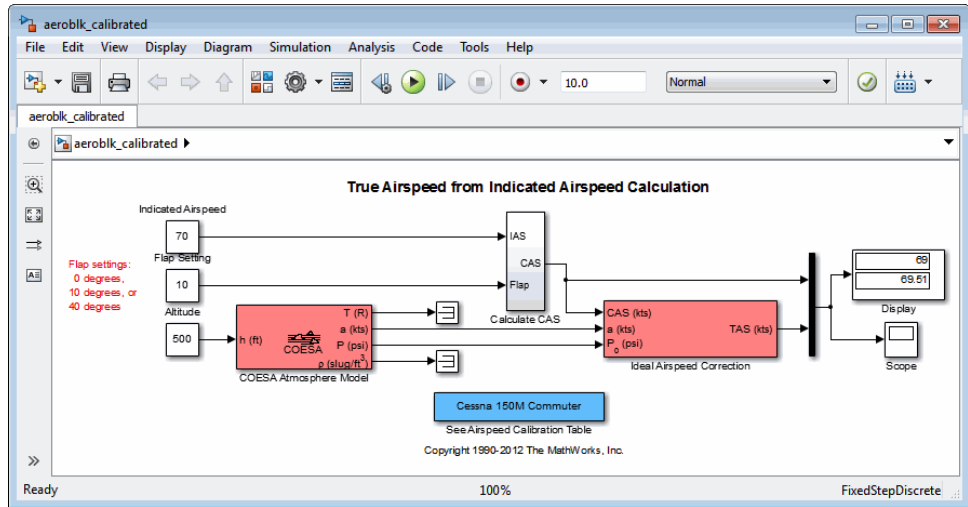
altitude as input and outputs the speed of sound and air pressure. Taking the speed of sound, air pressure, and airspeed as inputs, the Ideal Airspeed Correction block converts true airspeed to calibrated airspeed. Finally, the Calculate IAS subsystem uses the flap setting and calibrated airspeed to calculate indicated airspeed.

The model's Display block shows both indicated and calibrated airspeeds.



In the `aeroblk_calibrated` model, the aircraft is defined to be traveling at a constant speed of 70 knots (indicated airspeed) and altitude of 500 feet. The flaps are set to 10 degrees. The COESA Atmosphere Model block takes the altitude as input and outputs the speed of sound and air pressure. The Calculate CAS subsystem uses the flap setting and indicated airspeed to calculate the calibrated airspeed. Finally, using the speed of sound, air pressure, and true calibrated airspeed as inputs, the Ideal Airspeed Correction block converts calibrated airspeed back to true airspeed.

The model's Display block shows both calibrated and true airspeeds.



## 1903 Wright Flyer

In this section...
“Introduction” on page 3-10
“Wright Flyer Model” on page 3-11
“Airframe Subsystem” on page 3-11
“Environment Subsystem” on page 3-15
“Pilot Subsystem” on page 3-16
“Run the Simulation” on page 3-17
“References” on page 3-19

### Introduction

---

**Note** The final section of this study requires the Simulink 3D Animation software.

---

This case study describes a model of the 1903 Wright Flyer. Built by Orville and Wilbur Wright, the Wright Flyer took to the skies in December 1903 and opened the age of controlled flight. The Wright brothers' flying machine achieved the following goals:

- Left the ground under its own power
- Moved forward and maintained its speed
- Landed at an elevation no lower than where it started

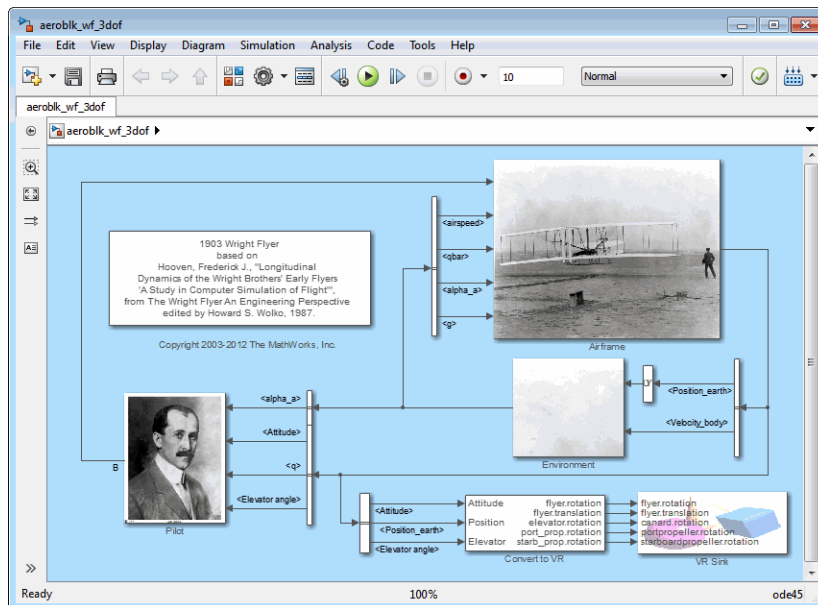
This model is based on an earlier simulation [1] that explored the longitudinal stability of the Wright Flyer and therefore modeled only forward and vertical motion along with the pitch angle. The Wright Flyer suffered from numerous engineering challenges, including dynamic and static instability. Laterally, the Flyer tended to overturn in crosswinds and gusts, and longitudinally, its pitch angle would undulate [2].

Under these constraints, the model recreates the longitudinal flight dynamics that pilots of the Wright Flyer would have experienced. Because they were able to control lateral motion, Orville and Wilbur Wright were able to maintain a relatively straight flight path.

Note, running this model generates assertion messages in the MATLAB Command Window. This is because the model illustrates the use of the Assertion block to indicate that the flyer is hitting the ground when landing.

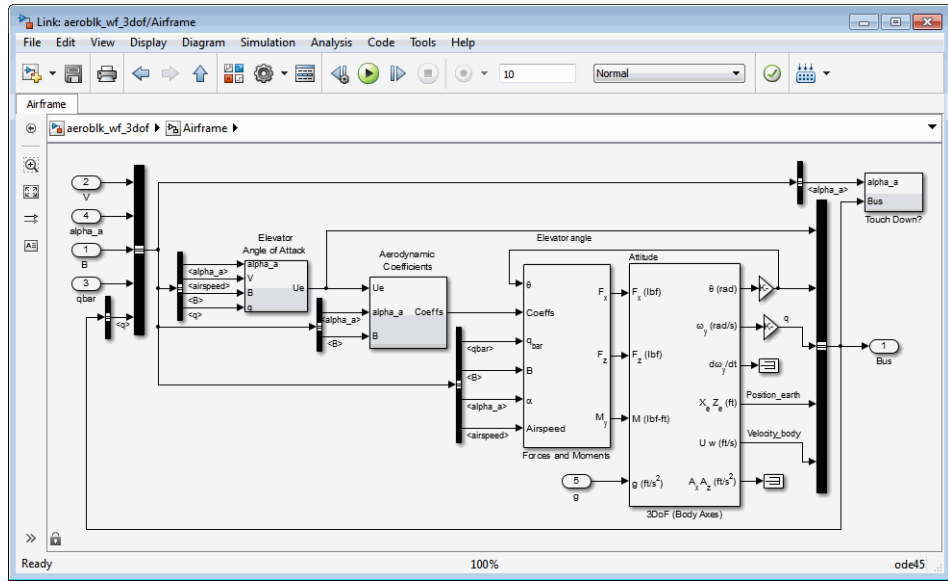
## Wright Flyer Model

Open the Wright Flyer model by entering `aeroblk_wf_3dof` at the MATLAB command line.



## Airframe Subsystem

The Airframe subsystem simulates the rigid body dynamics of the Wright Flyer airframe, including elevator angle of attack, aerodynamic coefficients, forces and moments, and three-degrees-of-freedom equations of motion.



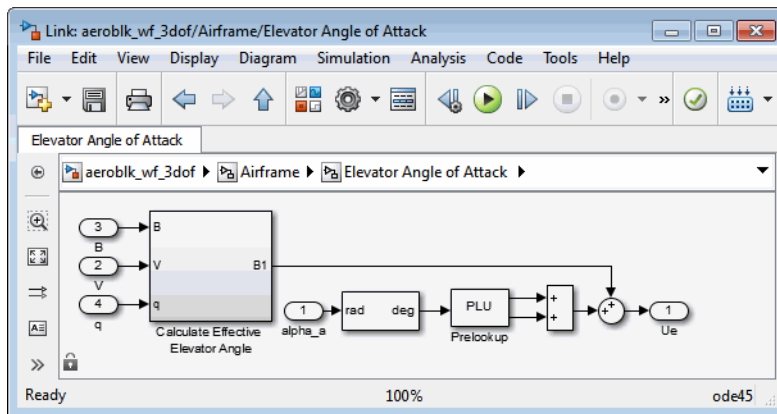
The Airframe subsystem consists of the following parts:

- “Elevator Angle of Attack Subsystem” on page 3-12
- “Aerodynamic Coefficients Subsystem” on page 3-13
- “Forces and Moments Subsystem” on page 3-14
- “3DoF (Body Axes) Block” on page 3-14

### Elevator Angle of Attack Subsystem

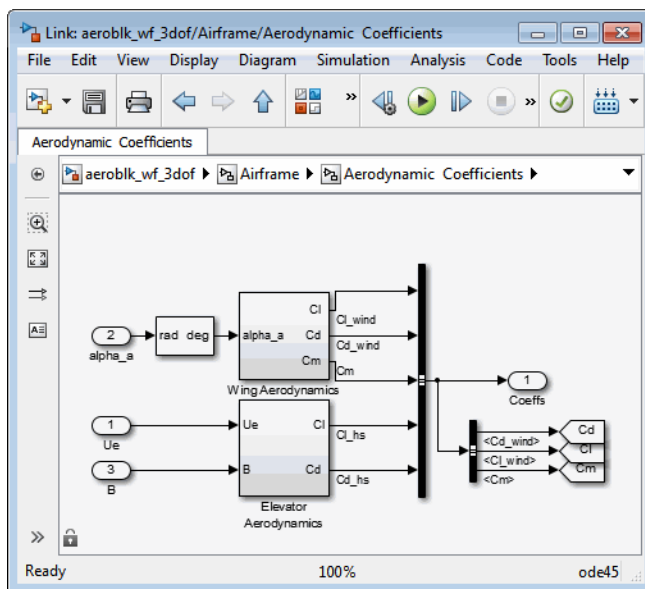
The Elevator Angle of Attack subsystem calculates the effective elevator angle for the Wright Flyer airframe and feeds its output to the Pilot subsystem.





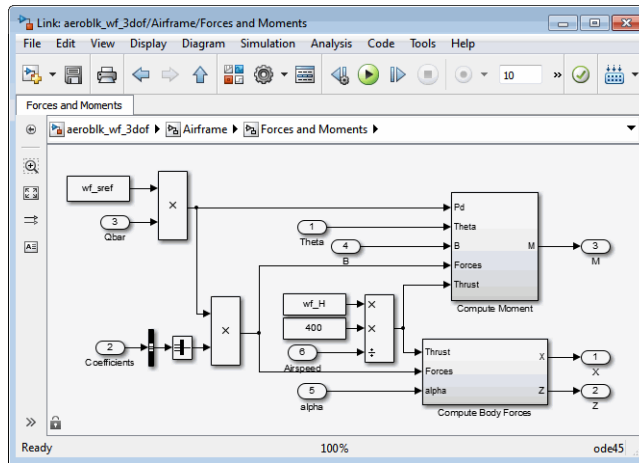
## Aerodynamic Coefficients Subsystem

The Aerodynamic Coefficients subsystem contains aerodynamic data and equations for calculating the aerodynamic coefficients, which are summed and passed to the Forces and Moments subsystem. Stored in data sets, the aerodynamic coefficients are determined by interpolation using Prelookup blocks.



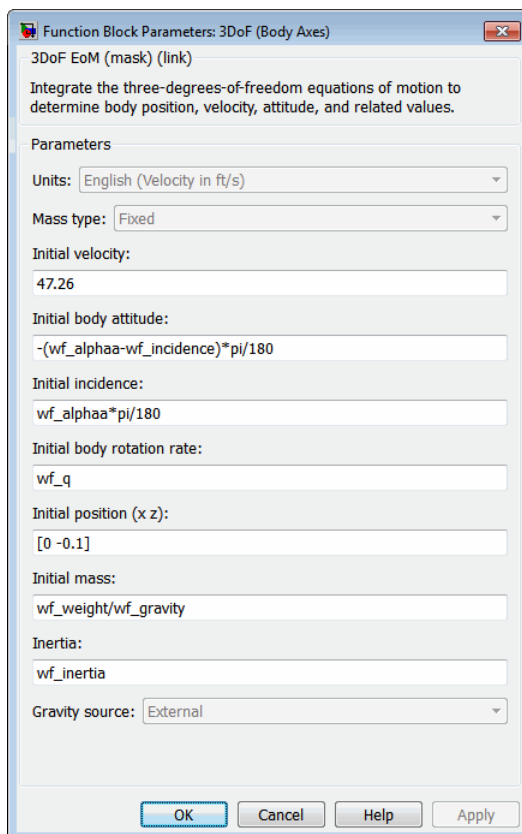
## Forces and Moments Subsystem

The aerodynamic forces and moments acting on the airframe are generated from aerodynamic coefficients. The Forces and Moments subsystem calculates the body forces and body moments acting on the airframe about the center of gravity. These forces and moments depend on the aerodynamic coefficients, thrust, dynamic pressure, and reference airframe parameters.



## 3DoF (Body Axes) Block

The 3DoF (Body Axes) block use equations of motion to define the linear and angular motion of the Wright Flyer airframe. It also performs conversions from the original model's axis system and the body axes.



### 3DoF (Body Axes) Block Parameters

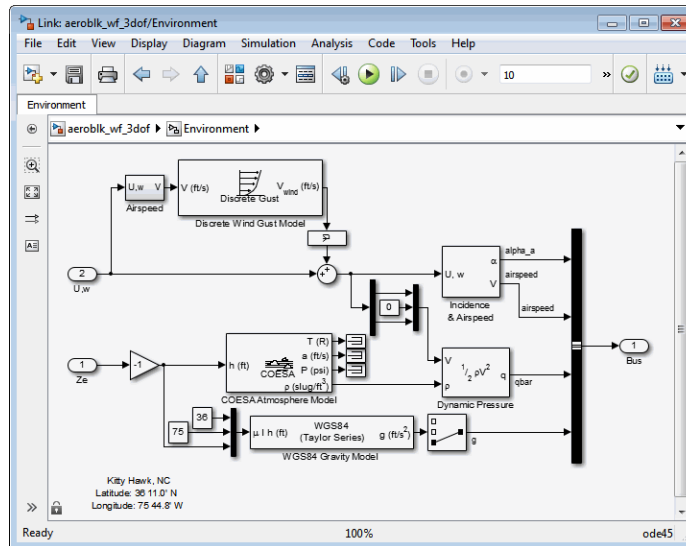
## Environment Subsystem

The first and final flights of the Wright Flyer occurred on December 17, 1903. Orville and Wilbur Wright chose an area near Kitty Hawk, North Carolina, situated near the Atlantic coast. Wind gusts of more than 25 miles per hour were recorded that day. After the final flight on that blustery December day, a wind gust caught and overturned the Wright Flyer, damaging it beyond repair.

The Environment subsystem of the Wright Flyer model contains a variety of blocks from the Environment sublibrary of the Aerospace Blockset software, including wind, atmosphere, and gravity, and calculates airspeed and

dynamic pressure. The Discrete Wind Gust Model block provides wind gusts to the simulated environment. The other blocks are

- The Incidence & Airspeed block calculates the angle of attack and airspeed.
- The COESA Atmosphere Model block calculates the air density.
- The Dynamic Pressure block computes the dynamic pressure from the air density and velocity.
- The WGS84 Gravity Model block produces the gravity at the Wright Flyer's latitude, longitude, and height.

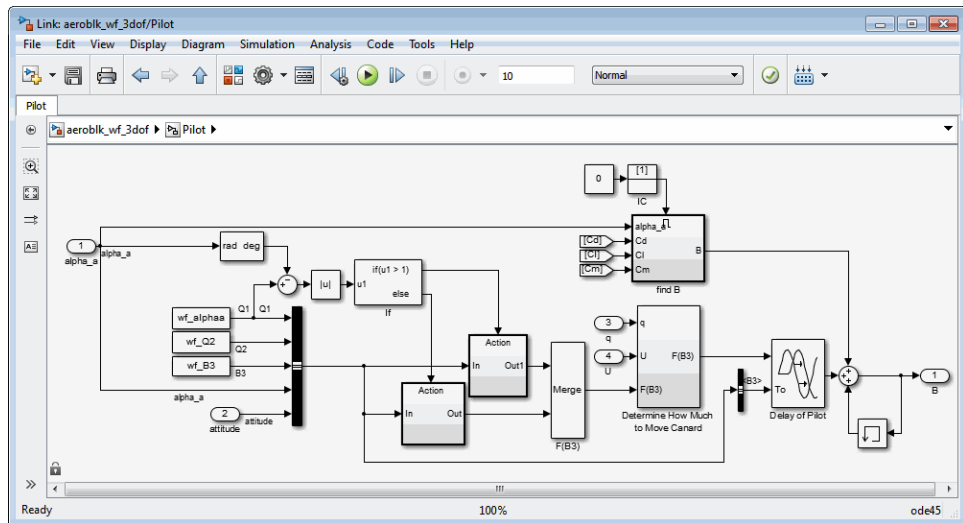


## Pilot Subsystem

The Pilot subsystem controls the aircraft by responding to both pitch angle (attitude) and angle of attack. If the angle of attack differs from the set angle of attack by more than one degree, the Pilot subsystem responds with a correction of the elevator (canard) angle. When the angular velocity exceeds  $\pm 0.02$  rad/s, angular velocity and angular acceleration are also taken into consideration with additional corrections to the elevator angle.

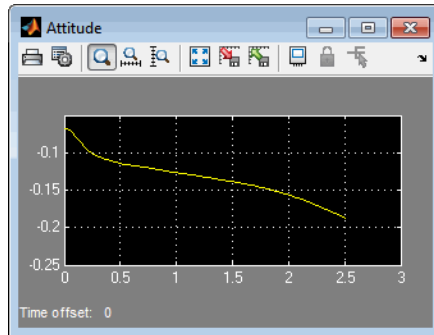
Pilot reaction time largely determined the success of the flights [1]. Without an automatic controller, a reaction time of 0.06 seconds is optimal for

successful flight. The Delay of Pilot (Variable Time Delay) block recreates this effect by producing a delay of no more than 0.08 second.



## Run the Simulation

The default values for this simulation allow the Wright Flyer model to take off and land successfully. The pilot reaction time (`wf_B3`) is set to 0.06 seconds, the desired angle of attack (`wf_alpha`) is constant, and the altitude attained is low. The Wright Flyer model reacts similarly to the actual Wright Flyer. It leaves the ground, moves forward, and lands on a point as high as that from which it started. This model exhibits the longitudinal undulation in attitude of the original aircraft.



#### **Attitude Scope (Measured in Radians)**

A pilot with quick reaction times and ideal flight conditions makes it possible to fly the Wright Flyer successfully. The Wright Flyer model confirms that controlling its longitudinal motion was a serious challenge. The longest recorded flight on that day lasted a mere 59 seconds and covered 852 feet.

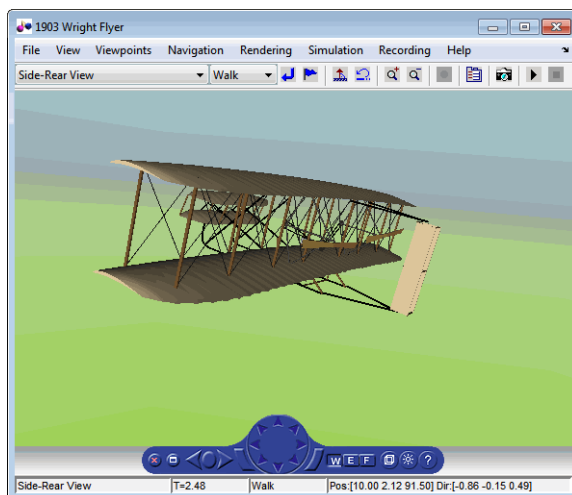
#### **Virtual Reality Visualization of the Wright Flyer**

---

**Note** This section requires the Simulink 3D Animation.

---

The Wright Flyer model also provides a virtual world visualization, coded in Virtual Reality Modeling Language (VRML) [3]. The VR Sink block in the main model allows you to view the flight motion in three dimensions.



### 1903 Wright Flyer Virtual Reality World

## References

- [1] Hooven, Frederick J., "Longitudinal Dynamics of the Wright Brothers' Early Flyers: A Study in Computer Simulation of Flight," from *The Wright Flyer: An Engineering Perspective*, ed. Howard S. Wolko, Smithsonian Institution Press, 1987.
- [2] Culick, F. E. C. and H. R. Jex, "Aerodynamics, Stability, and Control of the 1903 Wright Flyer," from *The Wright Flyer: An Engineering Perspective*, ed. Howard S. Wolko, Smithsonian Institution Press, 1987.
- [3] Thaddeus Beier created the initial Wright Flyer model in Inventor format, and Timothy Rohaly converted it to VRML.

## Additional Information About the 1903 Wright Flyer

- <http://www.wrightexperience.com>
- <http://wright.nasa.gov>

## NASA HL-20 Lifting Body Airframe

In this section...
“Introduction” on page 3-20
“NASA HL-20 Lifting Body” on page 3-20
“The HL-20 Airframe and Controller Model” on page 3-21
“References” on page 3-34

### Introduction

This case study models the airframe of a NASA HL-20 lifting body, a low-cost complement to the Space Shuttle orbiter. The HL-20 is unpowered, but the model includes both airframe and controller.

For most flight control designs, the airframe, or plant model, needs to be modeled, simulated, and analyzed. Ideally, this airframe should be modeled quickly, reusing blocks or model structure to reduce validation time and leave more time available for control design. In this study, the Aerospace Blockset software efficiently models portions of the HL-20 airframe. The remaining portions, including calculation of the aerodynamic coefficients, are modeled with the Simulink software. This case study examines the HL-20 airframe model and touches on how the aerodynamic data are used in the model.

### NASA HL-20 Lifting Body

The HL-20, also known as the Personnel Launch System (PLS), is a lifting body reentry vehicle designed to complement the Space Shuttle orbiter. It was developed originally as a low-cost solution for getting to and from low Earth orbit. It can carry up to 10 people and a limited cargo [1].

The HL-20 lifting body can be placed in orbit either by launching it vertically with booster rockets or by transporting it in the payload bay of the Space Shuttle orbiter. The HL-20 lifting body deorbits using a small onboard propulsion system. Its reentry profile is nose first, horizontal, and unpowered.





### **Top-Front View of the HL-20 Lifting Body (Photo: NASA Langley)**

The HL-20 design has a number of benefits:

- Rapid turnaround between landing and launch reduces operating costs.
- The HL-20 has exceptional flight safety.
- It can land conventionally on aircraft runways.

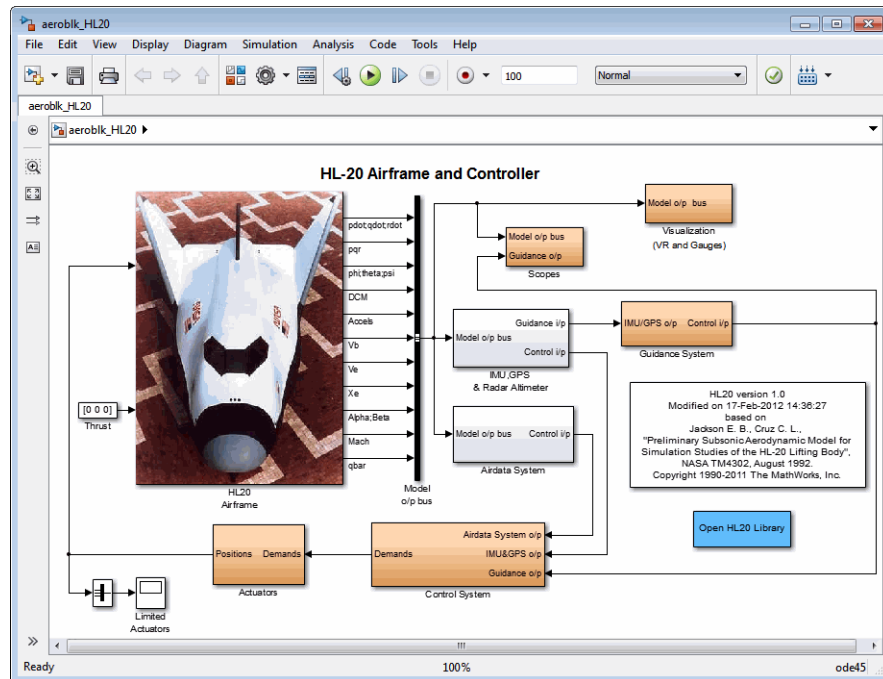
Potential uses for the HL-20 include

- Orbital rescue of stranded astronauts
- International Space Station crew exchanges
- Observation missions
- Satellite servicing missions

Although the HL-20 program is not currently active, the aerodynamic data from HL-20 tests are being used in current NASA projects [2].

### **The HL-20 Airframe and Controller Model**

You can open the HL-20 airframe and controller model by entering `aeroblk_HL20` at the MATLAB command line.



## Modeling Assumptions and Limitations

Preliminary aerodynamic data for the HL-20 lifting body are taken from NASA document TM4302 [1].

The airframe model incorporates several key assumptions and limitations:

- The airframe is assumed to be rigid and have constant mass, center of gravity, and inertia, since the model represents only the unpowered reentry portion of a mission.
- HL-20 is assumed to be a laterally symmetric vehicle.
- Compressibility (Mach) effects are assumed to be negligible.
- Control effectiveness is assumed to vary nonlinearly with angle of attack and linearly with angle of deflection. Control effectiveness is not dependent on sideslip angle.

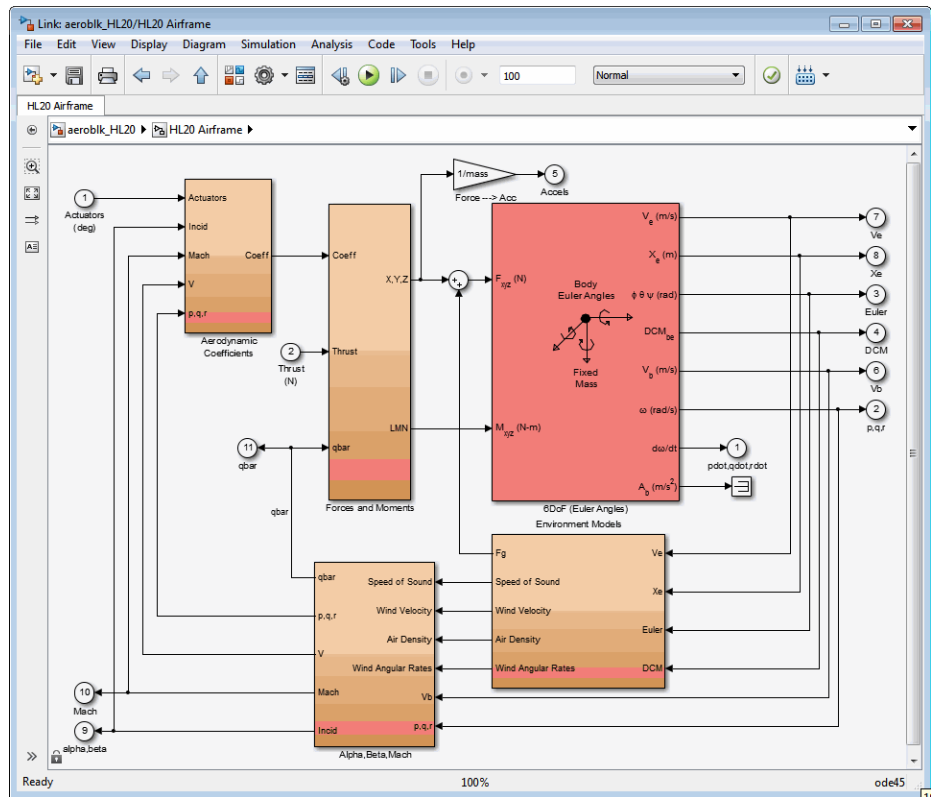
- The nonlinear six-degrees-of-freedom aerodynamic model is a representation of an early version of the HL-20. Therefore, the model is not intended for realistic performance simulation of later versions of the HL-20.

The typical airframe model consists of a number of components, such as

- Equations of motion
- Environmental models
- Calculation of aerodynamic coefficients, forces, and moments

The airframe subsystem of the HL-20 model contains five subsystems, which model the typical airframe components:

- “6DoF (Euler Angles) Subsystem” on page 3-24
- “Environmental Models Subsystem” on page 3-25
- “Alpha, Beta, Mach Subsystem” on page 3-27
- “Aerodynamic Coefficients Subsystem” on page 3-28
- “Forces and Moments Subsystem” on page 3-33



## HL-20 Airframe Subsystem

### 6DoF (Euler Angles) Subsystem

The 6DoF (Euler Angles) subsystem contains the six-degrees-of-freedom equations of motion for the airframe. In the 6DoF (Euler Angles) subsystem, the body attitude is propagated in time using an Euler angle representation. This subsystem is one of the equations of motion blocks from the Aerospace Blockset library. A quaternion representation is also available. See the 6DoF (Euler Angles) and 6DoF (Quaternion) block reference pages for more information on these blocks.

## Environmental Models Subsystem

The Environmental Models subsystem contains the following subsystems and blocks:

- The WGS84 Gravity Model block implements the mathematical representation of the geocentric equipotential ellipsoid of the World Geodetic System (WGS84).

See the WGS84 Gravity Model block reference page for more information on this block.

- The COESA Atmosphere Model block implements the mathematical representation of the 1976 Committee on Extension to the Standard Atmosphere (COESA) standard lower atmospheric values for absolute temperature, pressure, density, and speed of sound, given the input geopotential altitude.

See the COESA Atmosphere Model block reference page for more information on this block.

- The Wind Models subsystem contains the following blocks:

- The Wind Shear Model block adds wind shear to the model.

See the Wind Shear Model block reference page for more information on this block.

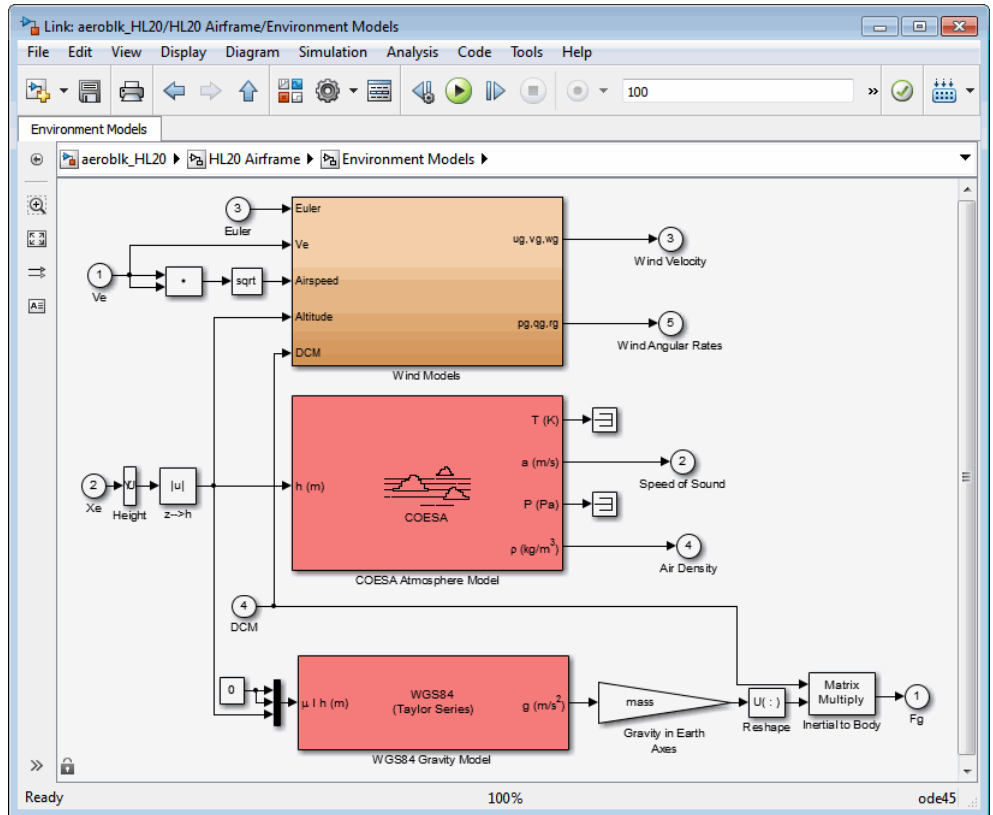
- The Discrete Wind Gust Model block implements a wind gust of the standard “1 - cosine” shape.

See the Discrete Wind Gust Model block reference page for more information on this block.

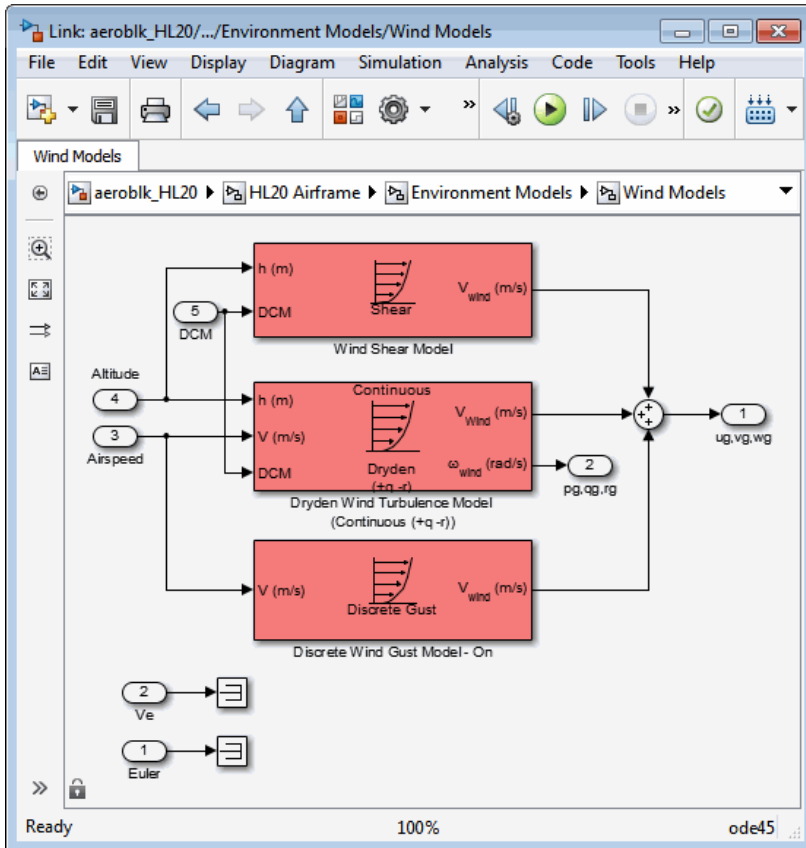
- The Dryden Wind Turbulence Model (Continuous) block uses the Dryden spectral representation to add turbulence to the aerospace model by passing band-limited white noise through appropriate forming filters.

See the Dryden Wind Turbulence Model (Continuous) block reference page for more information on this block.

The environmental models implement mathematical representations within standard references, such as U.S. Standard Atmosphere, 1976.



**Environmental Models in HL-20 Airframe Model**



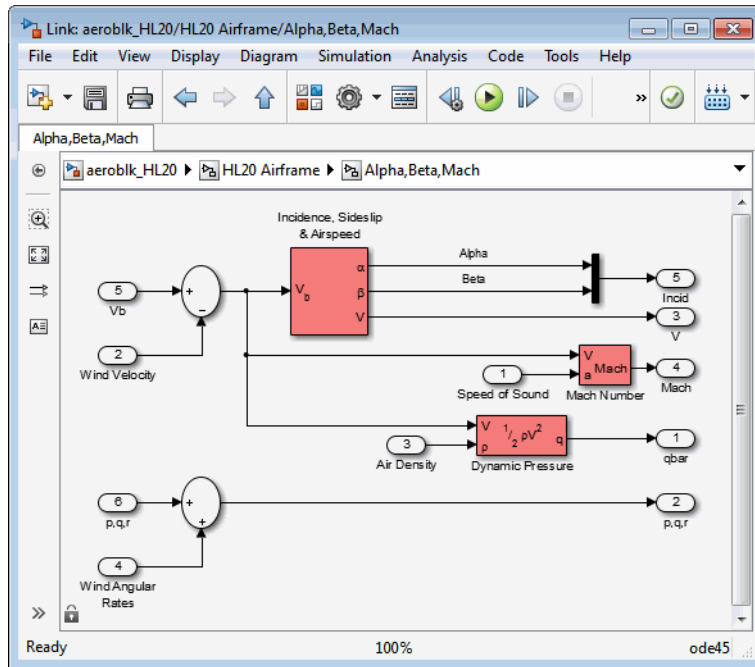
### Wind Models in HL-20 Airframe Model

#### Alpha, Beta, Mach Subsystem

The Alpha, Beta, Mach subsystem calculates additional parameters needed for the aerodynamic coefficient computation and lookup. These additional parameters include

- Mach number
- Incidence angles ( $\alpha, \beta$ )
- Airspeed
- Dynamic pressure

The Alpha, Beta, Mach subsystem corrects the body velocity for wind velocity and corrects the body rates for wind angular acceleration.



### Additional Computed Parameters for HL-20 Airframe Model (Alpha, Beta, Mach Subsystem)

### Aerodynamic Coefficients Subsystem

The Aerodynamic Coefficients subsystem contains aerodynamic data and equations for calculating the six aerodynamic coefficients, which are implemented as in reference [1]. The six aerodynamic coefficients follow.

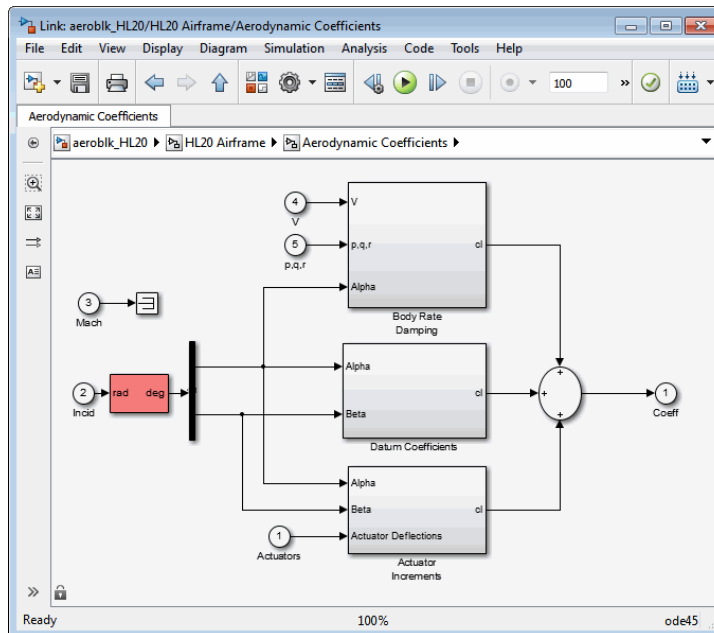
$C_x$	Axial-force coefficient
$C_y$	Side-force coefficient
$C_z$	Normal-force coefficient
$C_l$	Rolling-moment coefficient



$C_m$	Pitching-moment coefficient
$C_n$	Yawing-moment coefficient

Ground and landing gear effects are not included in this model.

The contribution of each of these coefficients is calculated in the subsystems (body rate, actuator increment, and datum), and then summed and passed to the Forces and Moments subsystem.



### Aerodynamic Coefficients in HL-20 Airframe Model

The aerodynamic data was gathered from wind tunnel tests, mainly on scaled models of a preliminary subsonic aerodynamic model of the HL-20. The data was curve fitted, and most of the aerodynamic coefficients are described by polynomial functions of angle of attack and sideslip angle. In-depth details about the aerodynamic data and the data reduction can be found in reference [1].

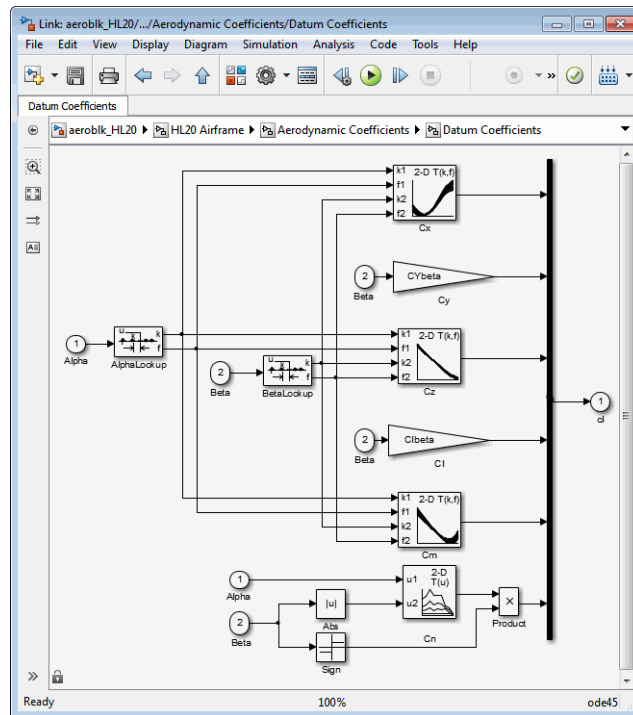
The polynomial functions contained in the `aeroblk_init_h120.m` file are used to calculate lookup tables used by the model's preload function. Lookup tables substitute for polynomial functions. Depending on the order and implementation of the function, using lookup tables can be more efficient than recalculating values at each time step with functions. To further improve efficiency, most tables are implemented as PreLook-up Index Search and Interpolation (n-D) using PreLook-up blocks. These blocks improve performance most when the model has a number of tables with identical breakpoints. These blocks reduce the number of times the model has to search for a breakpoint in a given time step. Once the tables are populated by the preload function, the aerodynamic coefficient can be computed.

The equations for calculating the six aerodynamic coefficients are divided among three subsystems:

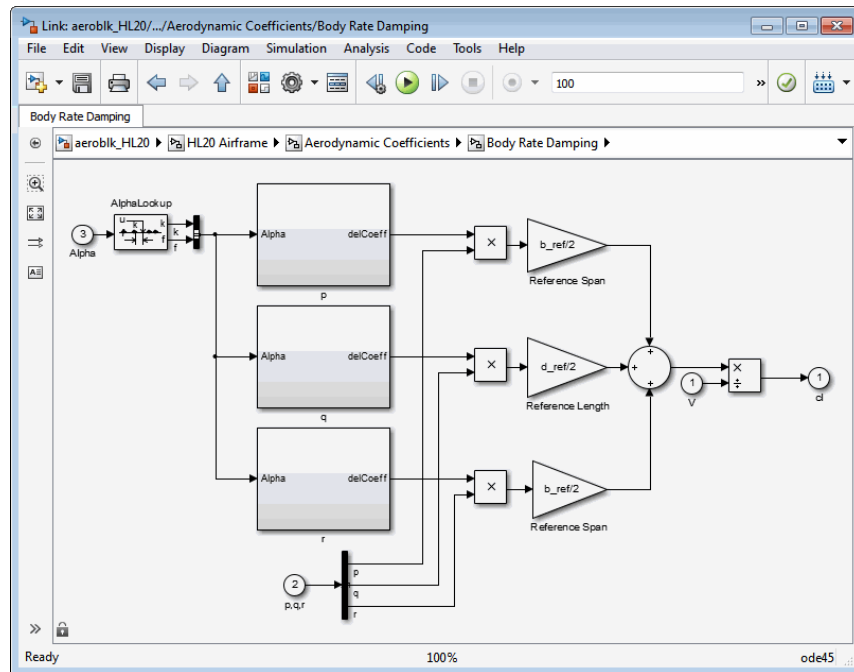
- “Datum Coefficients Subsystem” on page 3-30
- “Body Rate Damping Subsystem” on page 3-31
- “Actuator Increment Subsystem” on page 3-32

Summing the Datum Coefficients, Body Rate Damping, and Actuator Increments subsystem outputs generates the six aerodynamic coefficients used to calculate the airframe forces and moments [1].

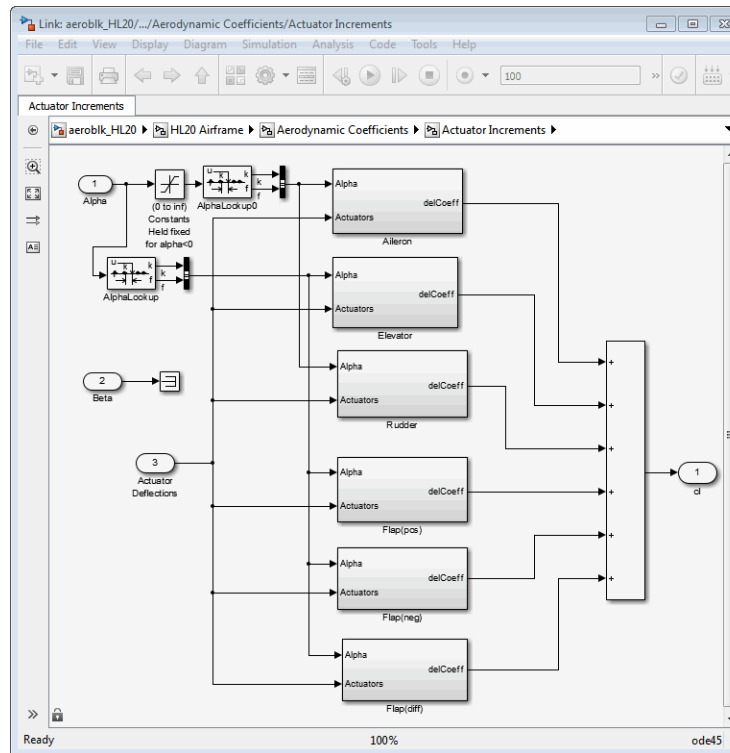
**Datum Coefficients Subsystem.** The Datum Coefficients subsystem calculates coefficients for the basic configuration without control surface deflection. These datum coefficients depend only on the incidence angles of the body.



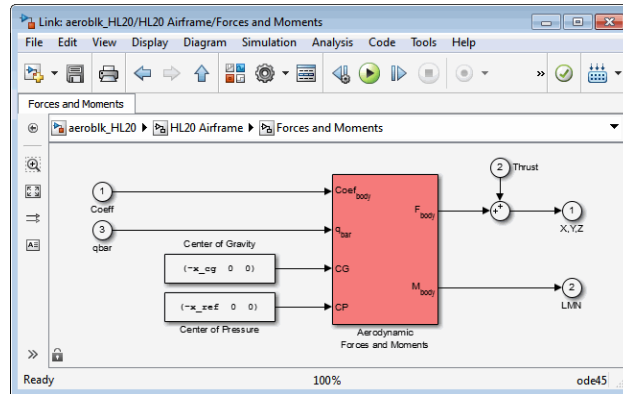
**Body Rate Damping Subsystem.** Dynamic motion derivatives are computed in the Body Rate Damping subsystem.



**Actuator Increment Subsystem.** Lookup tables determine the incremental changes to the coefficients due to the control surface deflections in the Actuator Increment subsystem. Available control surfaces include symmetric wing flaps (elevator), differential wing flaps (ailerons), positive body flaps, negative body flaps, differential body flaps, and an all-movable rudder.



**Forces and Moments Subsystem.** The Forces and Moments subsystem calculates the body forces and body moments acting on the airframe about the center of gravity. These forces and moments depend on the aerodynamic coefficients, thrust, dynamic pressure, and reference airframe parameters.



## Complete the Model

These subsystems that you have examined complete the HL-20 airframe. The next step in the flight control design process is to analyze, trim, and linearize the HL-20 airframe so that a flight control system can be designed for it. You can see an example of an auto-land flight control for the HL-20 airframe in the `aeroblk_HL20` example.

## References

[1] Jackson, E. B., and C. L. Cruz, "Preliminary Subsonic Aerodynamic Model for Simulation Studies of the HL-20 Lifting Body," NASA TM4302 (August 1992).

This document is included in the HL-20 Lifting Body .zip file available from MATLAB Central.

[2] Moring, F., Jr., "ISS 'Lifeboat' Study Includes ELVs," *Aviation Week & Space Technology* (May 20, 2002).

## Additional Information About the HL-20 Lifting Body

<http://www.astronautix.com/craft/hl20.htm>

# Blocks — Alphabetical List

---

# 1D Controller [A(v),B(v),C(v),D(v)]

---

**Purpose** Implement gain-scheduled state-space controller depending on one scheduling parameter

**Library** GNC/Control

**Description** The 1D Controller [A(v),B(v),C(v),D(v)] block implements a gain-scheduled state-space controller as defined by the equations



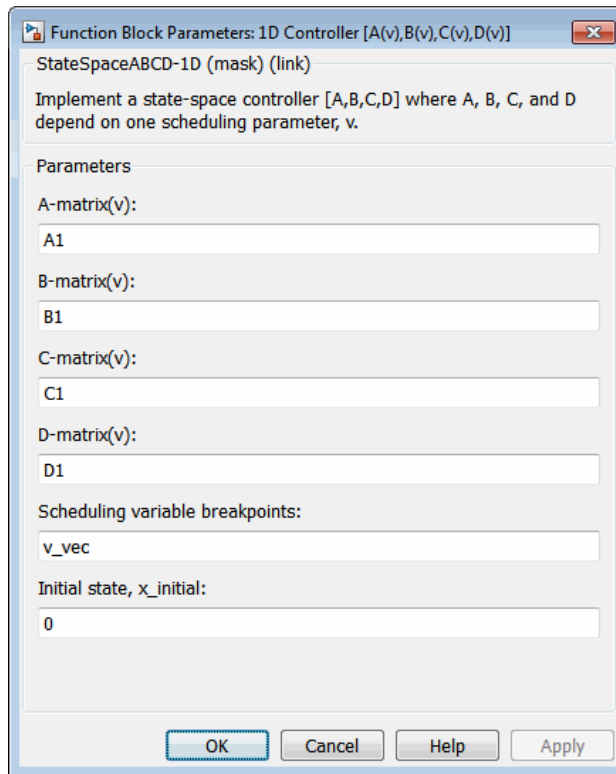
$$\dot{x} = A(v)x + B(v)y$$

$$u = C(v)x + D(v)y$$

where  $v$  is a parameter over which  $A$ ,  $B$ ,  $C$ , and  $D$  are defined. This type of controller scheduling assumes that the matrices  $A$ ,  $B$ ,  $C$ , and  $D$  vary smoothly as a function of  $v$ , which is often the case in aerospace applications.



## Dialog Box



### A-matrix(v)

*A*-matrix of the state-space implementation. In the case of 1-D scheduling, the *A*-matrix should have three dimensions, the last one corresponding to the scheduling variable *v*. Hence, for example, if the *A*-matrix corresponding to the first entry of *v* is the identity matrix, then  $A(:, :, 1) = [1 \ 0; 0 \ 1]$ ;

### B-matrix(v)

*B*-matrix of the state-space implementation. In the case of 1-D scheduling, the *B*-matrix should have three dimensions, the last one corresponding to the scheduling variable *v*. Hence, for

# 1D Controller $[A(v), B(v), C(v), D(v)]$

---

example, if the  $B$ -matrix corresponding to the first entry of  $v$  is the identity matrix, then  $B(:, :, 1) = [1 \ 0; 0 \ 1];$ .

## **C-matrix(v)**

$C$ -matrix of the state-space implementation. In the case of 1-D scheduling, the  $C$ -matrix should have three dimensions, the last one corresponding to the scheduling variable  $v$ . Hence, for example, if the  $C$ -matrix corresponding to the first entry of  $v$  is the identity matrix, then  $C(:, :, 1) = [1 \ 0; 0 \ 1];$ .

## **D-matrix(v)**

$D$ -matrix of the state-space implementation. In the case of 1-D scheduling, the  $D$ -matrix should have three dimensions, the last one corresponding to the scheduling variable  $v$ . Hence, for example, if the  $D$ -matrix corresponding to the first entry of  $v$  is the identity matrix, then  $D(:, :, 1) = [1 \ 0; 0 \ 1];$ .

## **Scheduling variable breakpoints**

Vector of the breakpoints for the scheduling variable. The length of  $v$  should be same as the size of the third dimension of  $A$ ,  $B$ ,  $C$ , and  $D$ .

## **Initial state, $x_{\text{initial}}$**

Vector of initial states for the controller, i.e., initial values for the state vector,  $x$ . It should have length equal to the size of the first dimension of  $A$ .

## **Inputs and Outputs**

<b>Input</b>	<b>Dimension Type</b>	<b>Description</b>
First	Any	Contains the measurements.
Second		Contains the scheduling variable conforming to the dimensions of the state-space matrices.

<b>Output</b>	<b>Dimension Type</b>	<b>Description</b>
First	Any	Contains the actuator demands.

## Assumptions and Limitations

If the scheduling parameter inputs to the block go out of range, then they are clipped; i.e., the state-space matrices are not interpolated out of range.

## Examples

See H-Infinity Controller (1 Dimensional Scheduling) in `aeroblk_lib_HL20` for an example of this block.

## See Also

1D Controller Blend  $u=(1-L).K1.y+L.K2.y$

1D Observer Form [A(v),B(v),C(v),F(v),H(v)]

1D Self-Conditioned [A(v),B(v),C(v),D(v)]

2D Controller [A(v),B(v),C(v),D(v)]

3D Controller [A(v),B(v),C(v),D(v)]

# 1D Controller Blend $u=(1-L).K1.y+L.K2.y$

**Purpose** Implement 1-D vector of state-space controllers by linear interpolation of their outputs

**Library** GNC/Control

## Description



The 1D Controller Blend  $u=(1-L).K1.y+L.K2.y$  block implements an array of state-space controller designs. The controllers are run in parallel, and their outputs interpolated according to the current flight condition or operating point. The advantage of this implementation approach is that the state-space matrices  $A$ ,  $B$ ,  $C$ , and  $D$  for the individual controller designs do not need to vary smoothly from one design point to the next.

For example, suppose two controllers are designed at two operating points  $v=v_{\min}$  and  $v=v_{\max}$ . The 1D Controller Blend block implements

$$\begin{aligned}\dot{x}_1 &= A_1x_1 + B_1y \\ u_1 &= C_1x_1 + D_1y \\ \dot{x}_2 &= A_2x_2 + B_2y \\ u_2 &= C_2x_2 + D_2y \\ u &= (1-\lambda)u_1 + \lambda u_2\end{aligned}$$

$$\lambda = \begin{cases} 0 & v < v_{\min} \\ \frac{v - v_{\min}}{v_{\max} - v_{\min}} & v_{\min} \leq v \leq v_{\max} \\ 1 & v > v_{\max} \end{cases}$$

For longer arrays of design points, the blocks only implement nearest neighbor designs. For the 1D Controller Blend block, at any given instant in time, three controller designs are being updated. This reduces computational requirements.

As the value of the scheduling parameter varies and the index of the controllers that need to be run changes, the states of the oncoming

# 1D Controller Blend $u=(1-L).K1.y+L.K2.y$

controller are initialized by using the self-conditioned form as defined for the Self-Conditioned [A,B,C,D] block.

## Dialog Box

Function Block Parameters: 1D Controller Blend:  $u=(1-L).K1.y+L.K2.y$

Blend-1D (mask) (link)

Blend between outputs of a 1-D vector of state-space controllers. All controllers must have the same state dimension.

Parameters

A-matrix(v):  
A1

B-matrix(v):  
B1

C-matrix(v):  
C1

D-matrix(v):  
D1

Scheduling variable breakpoints:  
[1 1.5 2]

Initial state,  $x_{initial}$ :  
0

Poles of  $A(v)-H(v)*C(v) = [w1 \dots wn]$ :  
[-5 -2]

OK Cancel Help Apply

### A-matrix(v)

A-matrix of the state-space implementation. In the case of 1-D blending, the A-matrix should have three dimensions, the last one corresponding to scheduling variable  $v$ . Hence, for example, if

# 1D Controller Blend $u=(1-L).K1.y+L.K2.y$

---

the  $A$ -matrix corresponding to the first entry of  $v$  is the identity matrix, then  $A(:, :, 1) = [1 \ 0; 0 \ 1];$ .

## **B-matrix(v)**

$B$ -matrix of the state-space implementation.

## **C-matrix(v)**

$C$ -matrix of the state-space implementation.

## **D-matrix(v)**

$D$ -matrix of the state-space implementation.

## **Scheduling variable breakpoints**

Vector of the breakpoints for the scheduling variable. The length of  $v$  should be same as the size of the third dimension of  $A$ ,  $B$ ,  $C$ , and  $D$ .

## **Initial state, x\_initial**

Vector of initial states for the controller, i.e., initial values for the state vector,  $x$ . It should have length equal to the size of the first dimension of  $A$ .

## **Poles of $A(v)-H(v)*C(v)$**

For oncoming controllers, an observer-like structure is used to ensure that the controller output tracks the current block output,  $u$ . The poles of the observer are defined in this dialog box as a vector, the number of poles being equal to the dimension of the  $A$ -matrix. Poles that are too fast result in sensor noise propagation, and poles that are too slow result in the failure of the controller output to track  $u$ .

## **Inputs and Outputs**

<b>Input</b>	<b>Dimension Type</b>	<b>Description</b>
First	Any	Contains the measurements.
Second		Contains the scheduling variable, conforming to the dimensions of the state-space matrices.

# 1D Controller Blend $u=(1-L).K1.y+L.K2.y$

Output	Dimension Type	Description
First	Any	Contains the actuator demands.

## Assumptions and Limitations

**Note** This block requires the Control System Toolbox™ product.

## Reference

Hyde, R. A., “H-infinity Aerospace Control Design - A VSTOL Flight Application,” Springer Verlag, *Advances in Industrial Control Series*, 1995. ISBN 3-540-19960-8. See Chapter 5.

## See Also

1D Controller [A(v),B(v),C(v),D(v)]  
1D Observer Form [A(v),B(v),C(v),F(v),H(v)]  
1D Self-Conditioned [A(v),B(v),C(v),D(v)]  
2D Controller Blend

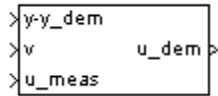
# 1D Observer Form $[A(v),B(v),C(v),F(v),H(v)]$

---

**Purpose** Implement gain-scheduled state-space controller in observer form depending on one scheduling parameter

**Library** GNC/Control

**Description** The 1D Observer Form  $[A(v),B(v),C(v),F(v),H(v)]$  block implements a gain-scheduled state-space controller defined in the following observer form:



$$\dot{x} = (A(v) + H(v)C(v))x + B(v)u_{meas} + H(v)(y - y_{dem})$$
$$u_{dem} = F(v)x$$

The main application of this block is to implement a controller designed using  $H$ -infinity loop-shaping, one of the design methods supported by Robust Control Toolbox.



# 1D Observer Form $[A(v), B(v), C(v), F(v), H(v)]$

## Dialog Box

Function Block Parameters: 1D Observer Form  $[A(v), B(v), C(v), F(v), H(v)]$

StateSpaceABCFH-1D (mask) (link)

Implement a state-space controller  $[A, B, C, F, H]$  in observer form where  $A$ ,  $B$ ,  $C$ ,  $F$ , and  $H$  depend on one scheduling parameter.

Parameters

A-matrix(v):  
A

B-matrix(v):  
B

C-matrix(v):  
C

F-matrix(v):  
F

H-matrix(v):  
H

Scheduling variable breakpoints:  
v\_vec

Initial state, x\_initial:  
0

OK Cancel Help Apply

### A-matrix(v)

A-matrix of the state-space implementation. The  $A$ -matrix should have three dimensions, the last one corresponding to the scheduling variable  $v$ . Hence, for example, if the  $A$ -matrix corresponding to the first entry of  $v$  is the identity matrix, then  $A(:, :, 1) = [1 \ 0; 0 \ 1];$

# 1D Observer Form $[A(v), B(v), C(v), F(v), H(v)]$

---

## **B-matrix(v)**

*B*-matrix of the state-space implementation. The *B*-matrix should have three dimensions, the last one corresponding to the scheduling variable  $v$ . Hence, for example, if the *B*-matrix corresponding to the first entry of  $v$  is the identity matrix, then  $B(:, :, 1) = [1 \ 0; 0 \ 1];$ .

## **C-matrix(v)**

*C*-matrix of the state-space implementation. The *C*-matrix should have three dimensions, the last one corresponding to the scheduling variable  $v$ . Hence, for example, if the *C*-matrix corresponding to the first entry of  $v$  is the identity matrix, then  $C(:, :, 1) = [1 \ 0; 0 \ 1];$ .

## **F-matrix(v)**

State-feedback matrix. The *F*-matrix should have three dimensions, the last one corresponding to the scheduling variable  $v$ . Hence, for example, if the *F*-matrix corresponding to the first entry of  $v$  is the identity matrix, then  $F(:, :, 1) = [1 \ 0; 0 \ 1];$ .

## **H-matrix(v)**

Observer (output injection) matrix. The *H*-matrix should have three dimensions, the last one corresponding to the scheduling variable  $v$ . Hence, for example, if the *H*-matrix corresponding to the first entry of  $v$  is the identity matrix, then  $H(:, :, 1) = [1 \ 0; 0 \ 1];$ .

## **Scheduling variable breakpoints**

Vector of the breakpoints for the scheduling variable. The length of  $v$  should be same as the size of the third dimension of *A*, *B*, *C*, *F*, and *H*.

## **Initial state, $x\_initial$**

Vector of initial states for the controller, i.e., initial values for the state vector,  $x$ . It should have length equal to the size of the first dimension of *A*.

# 1D Observer Form $[A(v),B(v),C(v),F(v),H(v)]$

## Inputs and Outputs

Input	Dimension	Type	Description
First			Contains the set-point error.
Second			Contains the scheduling variable.
Third			Contains the measured actuator position.

Output	Dimension	Type	Description
First			Contains the actuator demands.

## Assumptions and Limitations

If the scheduling parameter inputs to the block go out of range, then they are clipped; i.e., the state-space matrices are not interpolated out of range.

## Examples

See H-Infinity Controller (1 Dimensional Scheduling) in `aeroblk_lib_HL20` for an example of this block.

## Reference

Hyde, R. A., "H-infinity Aerospace Control Design - A VSTOL Flight Application," Springer Verlag, *Advances in Industrial Control Series*, 1995. ISBN 3-540-19960-8. See Chapter 6.

## See Also

1D Controller  $[A(v),B(v),C(v),D(v)]$   
1D Controller Blend  $u=(1-L).K1.y+L.K2.y$   
1D Self-Conditioned  $[A(v),B(v),C(v),D(v)]$   
2D Observer Form  $[A(v),B(v),C(v),F(v),H(v)]$   
3D Observer Form  $[A(v),B(v),C(v),F(v),H(v)]$

# 1D Self-Conditioned [A(v),B(v),C(v),D(v)]

---

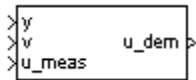
## Purpose

Implement gain-scheduled state-space controller in self-conditioned form depending on one scheduling parameter

## Library

GNC/Control

## Description



The 1D Self-Conditioned [A(v),B(v),C(v),D(v)] block implements a gain-scheduled state-space controller as defined by the equations

$$\dot{x} = A(v)x + B(v)y$$

$$u = C(v)x + D(v)y$$

in the self-conditioned form

$$\dot{z} = (A(v) - H(v)C(v))z + (B(v) - H(v)D(v))e + H(v)u_{meas}$$

$$u_{dem} = C(v)z + D(v)e$$

For the rationale behind this self-conditioned implementation, refer to the Self-Conditioned [A,B,C,D] block reference. This block implements a gain-scheduled version of the Self-Conditioned [A,B,C,D] block,  $v$  being the parameter over which  $A$ ,  $B$ ,  $C$ , and  $D$  are defined. This type of controller scheduling assumes that the matrices  $A$ ,  $B$ ,  $C$ , and  $D$  vary smoothly as a function of  $v$ , which is often the case in aerospace applications.

# 1D Self-Conditioned $[A(v),B(v),C(v),D(v)]$

## Dialog Box

Function Block Parameters: 1D Self-Conditioned  $[A(v),B(v),C(v),D(v)]$

StateSpaceSelfCond-1D (mask) (link)

Implement a state-space controller  $[A(v),B(v),C(v),D(v)]$  in a self-conditioned form. If  $u_{meas} = u_{dem}$ , then the implemented controller is  $[A,B,C,D]$ . If  $u_{meas}$  is limited, e.g., rate limiting, then the poles of the controller become those defined in the mask dialog box. Uses call to Control Systems Toolbox function `place.m` when initializing. A, B, C, and D should be 3-D matrices, the last dimension corresponding to the scheduling parameter, and the first two corresponding to the matrix for a given set of scheduling parameter values.

Parameters

A-matrix(v):  
A

B-matrix(v):  
B

C-matrix(v):  
C

D-matrix(v):  
D

Scheduling variable breakpoints:  
v\_vec

Initial state, x\_initial:  
0

Poles of  $A(v)-H(v)*C(v) = [w1 \dots wn]$ :  
[-5 -2]

OK Cancel Help Apply

### A-matrix(v)

A-matrix of the state-space implementation. The A-matrix should have three dimensions, the last one corresponding to

# 1D Self-Conditioned $[A(v), B(v), C(v), D(v)]$

---

the scheduling variable  $v$ . Hence, for example, if the  $A$ -matrix corresponding to the first entry of  $v$  is the identity matrix, then  $A(:, :, 1) = [1 \ 0; 0 \ 1];$ .

## **B-matrix(v)**

$B$ -matrix of the state-space implementation. The  $B$ -matrix should have three dimensions, the last one corresponding to the scheduling variable  $v$ . Hence, for example, if the  $B$ -matrix corresponding to the first entry of  $v$  is the identity matrix, then  $B(:, :, 1) = [1 \ 0; 0 \ 1];$ .

## **C-matrix(v)**

$C$ -matrix of the state-space implementation. The  $C$ -matrix should have three dimensions, the last one corresponding to the scheduling variable  $v$ . Hence, for example, if the  $C$ -matrix corresponding to the first entry of  $v$  is the identity matrix, then  $C(:, :, 1) = [1 \ 0; 0 \ 1];$ .

## **D-matrix(v)**

$D$ -matrix of the state-space implementation. The  $D$ -matrix should have three dimensions, the last one corresponding to the scheduling variable  $v$ . Hence, for example, if the  $D$ -matrix corresponding to the first entry of  $v$  is the identity matrix, then  $D(:, :, 1) = [1 \ 0; 0 \ 1];$ .

## **Scheduling variable breakpoints**

Vector of the breakpoints for the first scheduling variable. The length of  $v$  should be same as the size of the third dimension of  $A$ ,  $B$ ,  $C$ , and  $D$ .

## **Initial state, $x\_initial$**

Vector of initial states for the controller, i.e., initial values for the state vector,  $x$ . It should have length equal to the size of the first dimension of  $A$ .

## **Poles of $A(v)-H(v)*C(v)$**

Vector of the desired poles of  $A-HC$ . Note that the poles are assigned to the same locations for all values of the scheduling parameter  $v$ . Hence the number of pole locations defined should be equal to the length of the first dimension of the  $A$ -matrix.

# 1D Self-Conditioned $[A(v),B(v),C(v),D(v)]$

## Inputs and Outputs

Input	Dimension Type	Description
First	Any	Contains the measurements.
Second		Contains the scheduling variable, conforming to the dimensions of the state-space matrices.
Third		Contains the measured actuator position.

Output	Dimension Type	Description
First	Any	Contains the actuator demands.

## Assumptions and Limitations

If the scheduling parameter inputs to the block go out of range, then they are clipped; i.e., the state-space matrices are not interpolated out of range.

---

**Note** This block requires the Control System Toolbox product.

---

## Reference

The algorithm used to determine the matrix  $H$  is defined in Kautsky, Nichols, and Van Dooren, "Robust Pole Assignment in Linear State Feedback," *International Journal of Control*, Vol. 41, No. 5, pages 1129-1155, 1985.

## See Also

1D Controller  $[A(v),B(v),C(v),D(v)]$   
1D Controller Blend  $u=(1-L).K1.y+L.K2.y$   
1D Observer Form  $[A(v),B(v),C(v),F(v),H(v)]$   
2D Self-Conditioned  $[A(v),B(v),C(v),D(v)]$   
3D Self-Conditioned  $[A(v),B(v),C(v),D(v)]$

# 2D Controller [A(v),B(v),C(v),D(v)]

---

**Purpose** Implement gain-scheduled state-space controller depending on two scheduling parameters

**Library** GNC/Control

**Description** The 2D Controller [A(v),B(v),C(v),D(v)] block implements a gain-scheduled state-space controller as defined by the equations



$$\begin{aligned}\dot{x} &= A(v)x + B(v)y \\ u &= C(v)x + D(v)y\end{aligned}$$

where  $v$  is a vector of parameters over which  $A$ ,  $B$ ,  $C$ , and  $D$  are defined. This type of controller scheduling assumes that the matrices  $A$ ,  $B$ ,  $C$ , and  $D$  vary smoothly as a function of  $v$ , which is often the case in aerospace applications.



## Dialog Box

Function Block Parameters: 2D Controller [A(v),B(v),C(v),D(v)]

StateSpaceABCD-2D (mask) (link)

Implement a state-space controller [A,B,C,D] where A, B, C, and D depend on two scheduling parameters, v1 and v2.

Parameters

A-matrix(v1,v2):  
A

B-matrix(v1,v2):  
B

C-matrix(v1,v2):  
C

D-matrix(v1,v2):  
D

First scheduling variable (v1) breakpoints:  
v1\_vec

Second scheduling variable (v2) breakpoints:  
v2\_vec

Initial state, x\_initial:  
0

OK Cancel Help Apply

### A-matrix(v1,v2)

A-matrix of the state-space implementation. In the case of 2-D scheduling, the A-matrix should have four dimensions, the last two corresponding to scheduling variables  $v_1$  and  $v_2$ . Hence, for example, if the A-matrix corresponding to the first entry of  $v_1$  and first entry of  $v_2$  is the identity matrix, then  $A(:, :, 1, 1) = [1 \ 0; 0 \ 1];$ .

## 2D Controller $[A(v), B(v), C(v), D(v)]$

---

### **B-matrix(v1,v2)**

*B*-matrix of the state-space implementation. In the case of 2-D scheduling, the *B*-matrix should have four dimensions, the last two corresponding to scheduling variables  $v_1$  and  $v_2$ . Hence, for example, if the *B*-matrix corresponding to the first entry of  $v_1$  and first entry of  $v_2$  is the identity matrix, then  $B(:, :, 1, 1) = [1 \ 0; 0 \ 1];$ .

### **C-matrix(v1,v2)**

*C*-matrix of the state-space implementation. In the case of 2-D scheduling, the *C*-matrix should have four dimensions, the last two corresponding to scheduling variables  $v_1$  and  $v_2$ . Hence, for example, if the *C*-matrix corresponding to the first entry of  $v_1$  and first entry of  $v_2$  is the identity matrix, then  $C(:, :, 1, 1) = [1 \ 0; 0 \ 1];$ .

### **D-matrix(v1,v2)**

*D*-matrix of the state-space implementation. In the case of 2-D scheduling, the *D*-matrix should have four dimensions, the last two corresponding to scheduling variables  $v_1$  and  $v_2$ . Hence, for example, if the *D*-matrix corresponding to the first entry of  $v_1$  and first entry of  $v_2$  is the identity matrix, then  $D(:, :, 1, 1) = [1 \ 0; 0 \ 1];$ .

### **First scheduling variable (v1) breakpoints**

Vector of the breakpoints for the first scheduling variable. The length of  $v_1$  should be same as the size of the third dimension of *A*, *B*, *C*, and *D*.

### **Second scheduling variable (v2) breakpoints**

Vector of the breakpoints for the second scheduling variable. The length of  $v_2$  should be same as the size of the fourth dimension of *A*, *B*, *C*, and *D*.

### **Initial state, x\_initial**

Vector of initial states for the controller, i.e., initial values for the state vector,  $x$ . It should have length equal to the size of the first dimension of *A*.

# 2D Controller $[A(v),B(v),C(v),D(v)]$

## Inputs and Outputs

Input	Dimension Type	Description
First	Any	Contains the measurements.
Second		Contains the scheduling variable, conforming to the dimensions of the state-space matrices.
Third		Contains the scheduling variable, conforming to the dimensions of the state-space matrices.

Output	Dimension Type	Description
First	Any	Contains the actuator demands.

## Assumptions and Limitations

If the scheduling parameter inputs to the block go out of range, then they are clipped; i.e., the state-space matrices are not interpolated out of range.

## Examples

See H-Infinity Controller (Two Dimensional Scheduling) in `aerobk_lib_HL20` for an example of this block.

## See Also

1D Controller  $[A(v),B(v),C(v),D(v)]$   
2D Controller Blend  
2D Observer Form  $[A(v),B(v),C(v),F(v),H(v)]$   
2D Self-Conditioned  $[A(v),B(v),C(v),D(v)]$   
3D Controller  $[A(v),B(v),C(v),D(v)]$

# 2D Controller Blend

---

**Purpose** Implement 2-D vector of state-space controllers by linear interpolation of their outputs

**Library** GNC/Control

**Description**

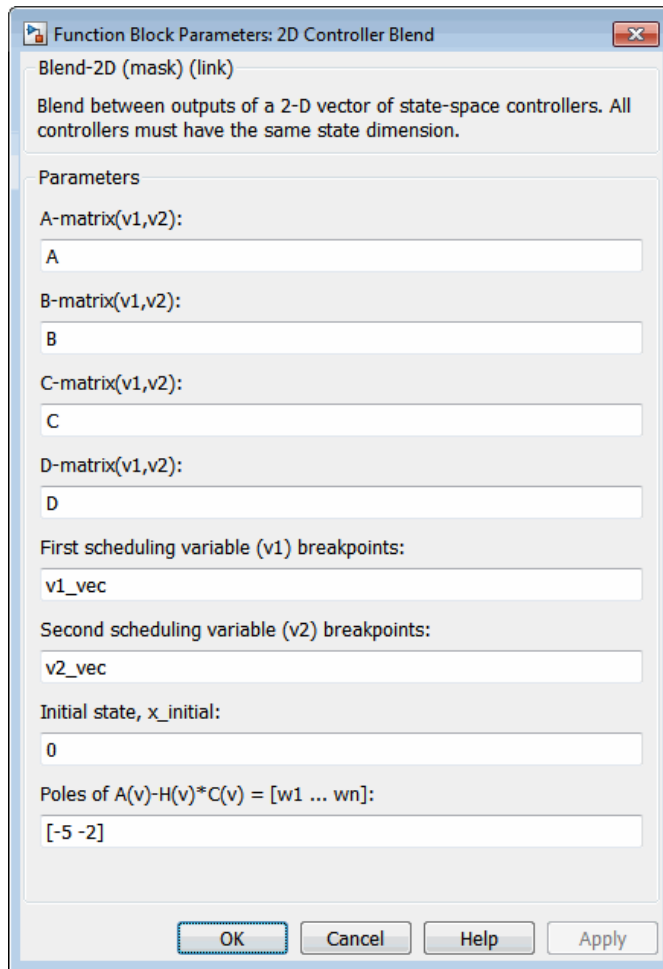


The 2D Controller Blend block implements an array of state-space controller designs. The controllers are run in parallel, and their outputs interpolated according to the current flight condition or operating point. The advantage of this implementation approach is that the state-space matrices  $A$ ,  $B$ ,  $C$ , and  $D$  for the individual controller designs do not need to vary smoothly from one design point to the next.

For the 2D Controller Blend block, at any given instant in time, nine controller designs are updated.

As the value of the scheduling parameter varies and the index of the controllers that need to be run changes, the states of the oncoming controller are initialized by using the self-conditioned form as defined for the Self-Conditioned [A,B,C,D] block.

## Dialog Box



### **A-matrix(v1,v2)**

A-matrix of the state-space implementation. In the case of 2-D blending, the A-matrix should have four dimensions, the last two corresponding to scheduling variables  $v1$  and  $v2$ . Hence, for example, if the A-matrix corresponding to the first entry of

## 2D Controller Blend

---

$v1$  and first entry of  $v2$  is the identity matrix, then  $A(:, :, 1, 1) = [1 \ 0; 0 \ 1];$ .

### **B-matrix(v1,v2)**

*B*-matrix of the state-space implementation.

### **C-matrix(v1,v2)**

*C*-matrix of the state-space implementation.

### **D-matrix(v1,v2)**

*D*-matrix of the state-space implementation.

### **First scheduling variable (v1) breakpoints**

Vector of the breakpoints for the first scheduling variable. The length of  $v1$  should be same as the size of the third dimension of *A*, *B*, *C*, and *D*.

### **Second scheduling variable (v2) breakpoints**

Vector of the breakpoints for the second scheduling variable. The length of  $v2$  should be same as the size of the fourth dimension of *A*, *B*, *C*, and *D*.

### **Initial state, x\_initial**

Vector of initial states for the controller, i.e., initial values for the state vector,  $x$ . It should have length equal to the size of the first dimension of *A*.

### **Poles of A(v)-H(v)\*C(v)**

For oncoming controllers, an observer-like structure is used to ensure that the controller output tracks the current block output,  $u$ . The poles of the observer are defined in this dialog box as a vector, the number of poles being equal to the dimension of the *A*-matrix. Poles that are too fast result in sensor noise propagation, and poles that are too slow result in the failure of the controller output to track  $u$ .

## Inputs and Outputs

Input	Dimension Type	Description
First	Any	Contains the measurements.
Second		Contains the scheduling variable, conforming to the dimensions of the state-space matrices.
Third		Contains the scheduling variable, conforming to the dimensions of the state-space matrices.

Output	Dimension Type	Description
First	Any	Contains the actuator demands.

## Assumptions and Limitations

**Note** This block requires the Control System Toolbox product.

## Reference

Hyde, R. A., "H-infinity Aerospace Control Design - A VSTOL Flight Application," Springer Verlag, *Advances in Industrial Control Series*, 1995. ISBN 3-540-19960-8. See Chapter 5.

## See Also

1D Controller Blend  $u=(1-L).K1.y+L.K2.y$

2D Controller  $[A(v),B(v),C(v),D(v)]$

2D Observer Form  $[A(v),B(v),C(v),F(v),H(v)]$

2D Self-Conditioned  $[A(v),B(v),C(v),D(v)]$

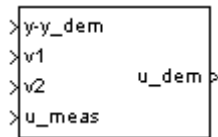
# 2D Observer Form $[A(v),B(v),C(v),F(v),H(v)]$

---

**Purpose** Implement gain-scheduled state-space controller in observer form depending on two scheduling parameters

**Library** GNC/Control

**Description** The 2D Observer Form  $[A(v),B(v),C(v),F(v),H(v)]$  block implements a gain-scheduled state-space controller defined in the following observer form:



$$\dot{x} = (A(v) + H(v)C(v))x + B(v)u_{meas} + H(v)(y - y_{dem})$$
$$u_{dem} = F(v)x$$

The main application of these blocks is to implement a controller designed using H-infinity loop-shaping, one of the design methods supported by Robust Control Toolbox.



# 2D Observer Form $[A(v),B(v),C(v),F(v),H(v)]$

## Dialog Box

Function Block Parameters: 2D Observer Form  $[A(v),B(v),C(v),F(v),H(v),H...]$

StateSpaceABCFH-2D (mask) (link)

Implement a state-space controller  $[A,B,C,F,H]$  in observer form where  $A$ ,  $B$ ,  $C$ ,  $F$ , and  $H$  depend on two scheduling parameters.

Parameters

A-matrix( $v1,v2$ ):  
A

B-matrix( $v1,v2$ ):  
B

C-matrix( $v1,v2$ ):  
C

F-matrix( $v1,v2$ ):  
F

H-matrix( $v1,v2$ ):  
H

First scheduling variable ( $v1$ ) breakpoints:  
v1\_vec

Second scheduling variable ( $v2$ ) breakpoints:  
v2\_vec

Initial state,  $x_{initial}$ :  
0

OK Cancel Help Apply

### **A-matrix( $v1,v2$ )**

A-matrix of the state-space implementation. In the case of 2-D scheduling, the  $A$ -matrix should have four dimensions, the last two corresponding to scheduling variables  $v1$  and  $v2$ . Hence, for example, if the  $A$ -matrix corresponding to the first entry of

## 2D Observer Form $[A(v), B(v), C(v), F(v), H(v)]$

---

$v1$  and first entry of  $v2$  is the identity matrix, then  $A(:, :, 1, 1) = [1 \ 0; 0 \ 1];$ .

### **B-matrix(v1,v2)**

$B$ -matrix of the state-space implementation. In the case of 2-D scheduling, the  $B$ -matrix should have four dimensions, the last two corresponding to scheduling variables  $v1$  and  $v2$ . Hence, for example, if the  $B$ -matrix corresponding to the first entry of  $v1$  and first entry of  $v2$  is the identity matrix, then  $B(:, :, 1, 1) = [1 \ 0; 0 \ 1];$ .

### **C-matrix(v1,v2)**

$C$ -matrix of the state-space implementation. In the case of 2-D scheduling, the  $C$ -matrix should have four dimensions, the last two corresponding to scheduling variables  $v1$  and  $v2$ . Hence, for example, if the  $C$ -matrix corresponding to the first entry of  $v1$  and first entry of  $v2$  is the identity matrix, then  $C(:, :, 1, 1) = [1 \ 0; 0 \ 1];$ .

### **F-matrix(v1,v2)**

State-feedback matrix. In the case of 2-D scheduling, the  $F$ -matrix should have four dimensions, the last two corresponding to scheduling variables  $v1$  and  $v2$ . Hence, for example, if the  $F$ -matrix corresponding to the first entry of  $v1$  and first entry of  $v2$  is the identity matrix, then  $F(:, :, 1, 1) = [1 \ 0; 0 \ 1];$ .

### **H-matrix(v1,v2)**

Observer (output injection) matrix. In the case of 2-D scheduling, the  $H$ -matrix should have four dimensions, the last two corresponding to scheduling variables  $v1$  and  $v2$ . Hence, for example, if the  $H$ -matrix corresponding to the first entry of  $v1$  and first entry of  $v2$  is the identity matrix, then  $H(:, :, 1, 1) = [1 \ 0; 0 \ 1];$ .

### **First scheduling variable (v1) breakpoints**

Vector of the breakpoints for the first scheduling variable. The length of  $v1$  should be same as the size of the third dimension of  $A$ ,  $B$ ,  $C$ ,  $F$ , and  $H$ .

# 2D Observer Form $[A(v), B(v), C(v), F(v), H(v)]$

## Second scheduling variable ( $v_2$ ) breakpoints

Vector of the breakpoints for the second scheduling variable. The length of  $v_2$  should be same as the size of the fourth dimension of  $A$ ,  $B$ ,  $C$ ,  $F$ , and  $H$ .

## Initial state, $x_{\text{initial}}$

Vector of initial states for the controller, i.e., initial values for the state vector,  $x$ . It should have length equal to the size of the first dimension of  $A$ .

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the set-point error.
Second		Contains the scheduling variable, conforming to the dimensions of the state-space matrices.
Third		Contains the scheduling variable, conforming to the dimensions of the state-space matrices.
Fourth		Contains the measured actuator position.

Output	Dimension Type	Description
First		Contains the actuator demands.

## Assumptions and Limitations

If the scheduling parameter inputs to the block go out of range, then they are clipped; i.e., the state-space matrices are not interpolated out of range.

## Examples

See H-Infinity Controller (Two Dimensional Scheduling) in `aeroblk_lib_HL20` for an example of this block.

## 2D Observer Form $[A(v),B(v),C(v),F(v),H(v)]$

---

### Reference

Hyde, R. A., "H-infinity Aerospace Control Design - A VSTOL Flight Application," Springer Verlag, *Advances in Industrial Control Series*, 1995. ISBN 3-540-19960-8. See Chapter 6.

### See Also

1D Controller  $[A(v),B(v),C(v),D(v)]$

2D Controller  $[A(v),B(v),C(v),D(v)]$

2D Controller Blend

2D Self-Conditioned  $[A(v),B(v),C(v),D(v)]$

3D Observer Form  $[A(v),B(v),C(v),F(v),H(v)]$

## 2D Self-Conditioned [A(v),B(v),C(v),D(v)]

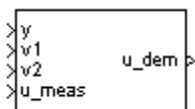
### Purpose

Implement gain-scheduled state-space controller in self-conditioned form depending on two scheduling parameters

### Library

GNC/Control

### Description



The 2D Self-Conditioned [A(v),B(v),C(v),D(v)] block implements a gain-scheduled state-space controller as defined by the equations

$$\begin{aligned}\dot{x} &= A(v)x + B(v)y \\ u &= C(v)x + D(v)y\end{aligned}$$

in the self-conditioned form

$$\begin{aligned}\dot{z} &= (A(v) - H(v)C(v))z + (B(v) - H(v)D(v))e + H(v)u_{meas} \\ u_{dem} &= C(v)z + D(v)e\end{aligned}$$

For the rationale behind this self-conditioned implementation, refer to the Self-Conditioned [A,B,C,D] block reference. This block implements a gain-scheduled version of the Self-Conditioned [A,B,C,D] block,  $v$  being the vector of parameters over which  $A$ ,  $B$ ,  $C$ , and  $D$  are defined. This type of controller scheduling assumes that the matrices  $A$ ,  $B$ ,  $C$ , and  $D$  vary smoothly as a function of  $v$ , which is often the case in aerospace applications.

## 2D Self-Conditioned $[A(v), B(v), C(v), D(v)]$

### Dialog Box

Function Block Parameters: 2D Self-Conditioned  $[A(v), B(v), C(v), D(v)]$

StateSpaceSelfCond-2D (mask) (link)

Implement a state-space controller  $[A(v_1, v_2), B(v_1, v_2), C(v_1, v_2), D(v_1, v_2)]$  in a self-conditioned form. If  $u_{\text{meas}} = u_{\text{dem}}$ , then the implemented controller is  $[A, B, C, D]$ . If  $u_{\text{meas}}$  is limited, e.g., rate limiting, then the poles of the controller become those defined in the mask dialog box. Uses call to Control Systems Toolbox function `place.m` when initializing. A, B, C, and D should be 4-D matrices, the last two dimensions corresponding to the scheduling parameters, and the first two corresponding to the matrix for a given set of scheduling parameter values.

Parameters

A-matrix( $v_1, v_2$ ):  
A

B-matrix( $v_1, v_2$ ):  
B

C-matrix( $v_1, v_2$ ):  
C

D-matrix( $v_1, v_2$ ):  
D

First scheduling variable ( $v_1$ ) breakpoints:  
v1\_vec

Second scheduling variable ( $v_2$ ) breakpoints:  
v2\_vec

Initial state,  $x_{\text{initial}}$ :  
0

Poles of  $A(v) - H(v) * C(v) = [w_1 \dots w_n]$ :  
[-5 -2]

OK Cancel Help Apply

### **A-matrix( $\mathbf{v}1, \mathbf{v}2$ )**

*A*-matrix of the state-space implementation. In the case of 2-D scheduling, the *A*-matrix should have four dimensions, the last two corresponding to scheduling variables  $v1$  and  $v2$ . Hence, for example, if the *A*-matrix corresponding to the first entry of  $v1$  and first entry of  $v2$  is the identity matrix, then  $A(:, :, 1, 1) = [1 \ 0; 0 \ 1];$ .

### **B-matrix( $\mathbf{v}1, \mathbf{v}2$ )**

*B*-matrix of the state-space implementation. In the case of 2-D scheduling, the *B*-matrix should have four dimensions, the last two corresponding to scheduling variables  $v1$  and  $v2$ . Hence, for example, if the *B*-matrix corresponding to the first entry of  $v1$  and first entry of  $v2$  is the identity matrix, then  $B(:, :, 1, 1) = [1 \ 0; 0 \ 1];$ .

### **C-matrix( $\mathbf{v}1, \mathbf{v}2$ )**

*C*-matrix of the state-space implementation. In the case of 2-D scheduling, the *C*-matrix should have four dimensions, the last two corresponding to scheduling variables  $v1$  and  $v2$ . Hence, for example, if the *C*-matrix corresponding to the first entry of  $v1$  and first entry of  $v2$  is the identity matrix, then  $C(:, :, 1, 1) = [1 \ 0; 0 \ 1];$ .

### **D-matrix( $\mathbf{v}1, \mathbf{v}2$ )**

*D*-matrix of the state-space implementation. In the case of 2-D scheduling, the *D*-matrix should have four dimensions, the last two corresponding to scheduling variables  $v1$  and  $v2$ . Hence, for example, if the *D*-matrix corresponding to the first entry of  $v1$  and first entry of  $v2$  is the identity matrix, then  $D(:, :, 1, 1) = [1 \ 0; 0 \ 1];$ .

### **First scheduling variable ( $\mathbf{v}1$ ) breakpoints**

Vector of the breakpoints for the first scheduling variable. The length of  $v1$  should be same as the size of the third dimension of *A*, *B*, *C*, and *D*.

## 2D Self-Conditioned $[A(v), B(v), C(v), D(v)]$

### Second scheduling variable ( $v_2$ ) breakpoints

Vector of the breakpoints for the second scheduling variable. The length of  $v_2$  should be same as the size of the fourth dimension of  $A$ ,  $B$ ,  $C$ , and  $D$ .

### Initial state, $x\_initial$

Vector of initial states for the controller, i.e., initial values for the state vector,  $x$ . It should have length equal to the size of the first dimension of  $A$ .

### Poles of $A(v)-H(v)*C(v)$

Vector of the desired poles of  $A-HC$ . Note that the poles are assigned to the same locations for all values of the scheduling parameter,  $v$ . Hence, the number of pole locations defined should be equal to the length of the first dimension of the  $A$ -matrix.

## Inputs and Outputs

Input	Dimension	Type	Description
First			Contains the measurements.
Second			Contains the scheduling variable, conforming to the dimensions of the state-space matrices.
Third			Contains the scheduling variable, conforming to the dimensions of the state-space matrices.
Fourth			Contains the measured actuator position.

Output	Dimension	Type	Description
First			Contains the actuator demands.



## 2D Self-Conditioned $[A(v),B(v),C(v),D(v)]$

---

### Assumptions and Limitations

If the scheduling parameter inputs to the block go out of range, then they are clipped; i.e., the state-space matrices are not interpolated out of range.

---

**Note** This block requires the Control System Toolbox product.

---

### Reference

The algorithm used to determine the matrix H is defined in Kautsky, Nichols, and Van Dooren, "Robust Pole Assignment in Linear State Feedback," *International Journal of Control*, Vol. 41, No. 5, pages 1129-1155, 1985.

### See Also

1D Self-Conditioned  $[A(v),B(v),C(v),D(v)]$

2D Controller  $[A(v),B(v),C(v),D(v)]$

2D Controller Blend

2D Observer Form  $[A(v),B(v),C(v),F(v),H(v)]$

3D Self-Conditioned  $[A(v),B(v),C(v),D(v)]$

# 3D Controller [A(v),B(v),C(v),D(v)]

---

**Purpose** Implement gain-scheduled state-space controller depending on three scheduling parameters

**Library** GNC/Control

**Description** The 3D Controller [A(v),B(v),C(v),D(v)] block implements a gain-scheduled state-space controller as defined by the equations



$$\begin{aligned}\dot{x} &= A(v)x + B(v)u \\ u &= C(v)x + D(v)y\end{aligned}$$

where  $v$  is a vector of parameters over which  $A$ ,  $B$ ,  $C$ , and  $D$  are defined. This type of controller scheduling assumes that the matrices  $A$ ,  $B$ ,  $C$ , and  $D$  vary smoothly as a function of  $v$ , which is often the case in aerospace applications.

## Dialog Box

Function Block Parameters: 2D Controller [A(v),B(v),C(v),D(v)]

StateSpaceABCD-2D (mask) (link)

Implement a state-space controller [A,B,C,D] where A, B, C, and D depend on two scheduling parameters, v1 and v2.

Parameters

A-matrix(v1,v2):  
A

B-matrix(v1,v2):  
B

C-matrix(v1,v2):  
C

D-matrix(v1,v2):  
D

First scheduling variable (v1) breakpoints:  
v1\_vec

Second scheduling variable (v2) breakpoints:  
v2\_vec

Initial state, x\_initial:  
0

OK Cancel Help Apply

### A-matrix(v1,v2,v3)

A-matrix of the state-space implementation. In the case of 3-D scheduling, the A-matrix should have five dimensions, the last three corresponding to scheduling variables  $v1$ ,  $v2$ , and  $v3$ . Hence, for example, if the A-matrix corresponding to the first entry of  $v1$ , the first entry of  $v2$ , and the first entry of  $v3$  is the identity matrix, then  $A(:, :, 1, 1, 1) = [1 \ 0; 0 \ 1];$ .

## 3D Controller $[A(v), B(v), C(v), D(v)]$

---

### **B-matrix(v1,v2,v3)**

*B*-matrix of the state-space implementation. In the case of 3-D scheduling, the *B*-matrix should have five dimensions, the last three corresponding to scheduling variables  $v_1$ ,  $v_2$ , and  $v_3$ . Hence, for example, if the *B*-matrix corresponding to the first entry of  $v_1$ , the first entry of  $v_2$ , and the first entry of  $v_3$  is the identity matrix, then  $B(:, :, 1, 1, 1) = [1 \ 0; 0 \ 1];$ .

### **C-matrix(v1,v2,v3)**

*C*-matrix of the state-space implementation. In the case of 3-D scheduling, the *C*-matrix should have five dimensions, the last three corresponding to scheduling variables  $v_1$ ,  $v_2$ , and  $v_3$ . Hence, for example, if the *C*-matrix corresponding to the first entry of  $v_1$ , the first entry of  $v_2$ , and the first entry of  $v_3$  is the identity matrix, then  $C(:, :, 1, 1, 1) = [1 \ 0; 0 \ 1];$ .

### **D-matrix(v1,v2,v3)**

*D*-matrix of the state-space implementation. In the case of 3-D scheduling, the *D*-matrix should have five dimensions, the last three corresponding to scheduling variables  $v_1$ ,  $v_2$ , and  $v_3$ . Hence, for example, if the *D*-matrix corresponding to the first entry of  $v_1$ , the first entry of  $v_2$ , and the first entry of  $v_3$  is the identity matrix, then  $D(:, :, 1, 1, 1) = [1 \ 0; 0 \ 1];$ .

### **First scheduling variable (v1) breakpoints**

Vector of the breakpoints for the first scheduling variable. The length of  $v_1$  should be same as the size of the third dimension of *A*, *B*, *C*, and *D*.

### **Second scheduling variable (v2) breakpoints**

Vector of the breakpoints for the second scheduling variable. The length of  $v_2$  should be same as the size of the fourth dimension of *A*, *B*, *C*, and *D*.

### **Third scheduling variable (v3) breakpoints**

Vector of the breakpoints for the third scheduling variable. The length of  $v_3$  should be same as the size of the fifth dimension of *A*, *B*, *C*, and *D*.

## Initial state, $x\_initial$

Vector of initial states for the controller, i.e., initial values for the state vector,  $x$ . It should have length equal to the size of the first dimension of  $A$ .

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the measurements.
Second		Contains the scheduling variable, ordered conforming to the dimensions of the state-space matrices.
Third		Contains the scheduling variable, ordered conforming to the dimensions of the state-space matrices.
Fourth		Contains the scheduling variable, ordered conforming to the dimensions of the state-space matrices.

Output	Dimension Type	Description
First		Contains the actuator demands.

## Assumptions and Limitations

If the scheduling parameter input to the block go out of range, then they are clipped; i.e., the state-space matrices are not interpolated out of range.

## See Also

- 1D Controller  $[A(v),B(v),C(v),D(v)]$
- 2D Controller  $[A(v),B(v),C(v),D(v)]$
- 3D Observer Form  $[A(v),B(v),C(v),F(v),H(v)]$
- 3D Self-Conditioned  $[A(v),B(v),C(v),D(v)]$

# 3D Observer Form $[A(v),B(v),C(v),F(v),H(v)]$

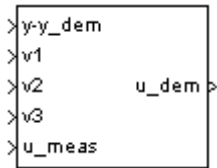
## Purpose

Implement gain-scheduled state-space controller in observer form depending on three scheduling parameters

## Library

GNC/Control

## Description



The 3D Observer Form  $[A(v),B(v),C(v),F(v),H(v)]$  block implements a gain-scheduled state-space controller defined in the following observer form:

$$\dot{x} = (A(v) + H(v)C(v))x + B(v)u_{meas} + H(v)(y - y_{dem})$$
$$u_{dem} = F(v)x$$

The main application of this block is to implement a controller designed using H-infinity loop-shaping, one of the design methods supported by Robust Control Toolbox.

# 3D Observer Form $[A(v), B(v), C(v), F(v), H(v)]$

## Dialog Box

Function Block Parameters: 3D Observer Form  $[A(v), B(v), C(v), F(v), H(v)]$

StateSpaceABCFH-3D (mask) (link)

Implement a state-space controller  $[A, B, C, F, H]$  in observer form where  $A$ ,  $B$ ,  $C$ ,  $F$ , and  $H$  depend on three scheduling parameters.

Parameters

A-matrix  $(v_1, v_2, v_3)$ :  
A

B-matrix  $(v_1, v_2, v_3)$ :  
B

C-matrix  $(v_1, v_2, v_3)$ :  
C

F-matrix  $(v_1, v_2, v_3)$ :  
F

H-matrix  $(v_1, v_2, v_3)$ :  
H

First scheduling variable ( $v_1$ ) breakpoints:  
v1\_vec

Second scheduling variable ( $v_2$ ) breakpoints:  
v2\_vec

Third scheduling variable ( $v_3$ ) breakpoints:  
v3\_vec

Initial state,  $x_{initial}$ :  
0

OK Cancel Help Apply

## 3D Observer Form $[A(v), B(v), C(v), F(v), H(v)]$

---

### **A-matrix(v1,v2,v3)**

*A*-matrix of the state-space implementation. In the case of 3-D scheduling, the *A*-matrix should have five dimensions, the last three corresponding to scheduling variables  $v_1$ ,  $v_2$ , and  $v_3$ . Hence, for example, if the *A*-matrix corresponding to the first entry of  $v_1$ , the first entry of  $v_2$ , and the first entry of  $v_3$  is the identity matrix, then  $A(:, :, 1, 1, 1) = [1 \ 0; 0 \ 1];$ .

### **B-matrix(v1,v2,v3)**

*B*-matrix of the state-space implementation. In the case of 3-D scheduling, the *B*-matrix should have five dimensions, the last three corresponding to scheduling variables  $v_1$ ,  $v_2$ , and  $v_3$ . Hence, for example, if the *B*-matrix corresponding to the first entry of  $v_1$ , the first entry of  $v_2$ , and the first entry of  $v_3$  is the identity matrix, then  $B(:, :, 1, 1, 1) = [1 \ 0; 0 \ 1];$ .

### **C-matrix(v1,v2,v3)**

*C*-matrix of the state-space implementation. In the case of 3-D scheduling, the *C*-matrix should have five dimensions, the last three corresponding to scheduling variables  $v_1$ ,  $v_2$ , and  $v_3$ . Hence, for example, if the *C*-matrix corresponding to the first entry of  $v_1$ , the first entry of  $v_2$ , and the first entry of  $v_3$  is the identity matrix, then  $C(:, :, 1, 1, 1) = [1 \ 0; 0 \ 1];$ .

### **F-matrix(v1,v2,v3)**

State-feedback matrix. In the case of 3-D scheduling, the *F*-matrix should have five dimensions, the last three corresponding to scheduling variables  $v_1$ ,  $v_2$ , and  $v_3$ . Hence, for example, if the *F*-matrix corresponding to the first entry of  $v_1$ , the first entry of  $v_2$ , and the first entry of  $v_3$  is the identity matrix, then  $F(:, :, 1, 1, 1) = [1 \ 0; 0 \ 1];$ .

### **H-matrix(v1,v2,v3)**

Observer (output injection) matrix. In the case of 3-D scheduling, the *H*-matrix should have five dimensions, the last three corresponding to scheduling variables  $v_1$ ,  $v_2$ , and  $v_3$ . Hence, for example, if the *H*-matrix corresponding to the first entry of  $v_1$ , the first entry of  $v_2$ , and the first entry of  $v_3$  is the identity matrix, then  $H(:, :, 1, 1, 1) = [1 \ 0; 0 \ 1];$ .



# 3D Observer Form $[A(v), B(v), C(v), F(v), H(v)]$

## First scheduling variable ( $v_1$ ) breakpoints

Vector of the breakpoints for the first scheduling variable. The length of  $v_1$  should be same as the size of the third dimension of  $A$ ,  $B$ ,  $C$ ,  $F$ , and  $H$ .

## Second scheduling variable ( $v_2$ ) breakpoints

Vector of the breakpoints for the second scheduling variable. The length of  $v_2$  should be same as the size of the fourth dimension of  $A$ ,  $B$ ,  $C$ ,  $F$ , and  $H$ .

## Third scheduling variable ( $v_3$ ) breakpoints

Vector of the breakpoints for the third scheduling variable. The length of  $v_3$  should be same as the size of the fifth dimension of  $A$ ,  $B$ ,  $C$ ,  $F$ , and  $H$ .

## Initial state, $x_{\text{initial}}$

Vector of initial states for the controller, i.e., initial values for the state vector,  $x$ . It should have length equal to the size of the first dimension of  $A$ .

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the set-point error.
Second		Contains the scheduling variable, ordered conforming to the dimensions of the state-space matrices.
Third		Contains the scheduling variable, ordered conforming to the dimensions of the state-space matrices.
Fourth		Contains the scheduling variable, ordered conforming to the dimensions of the state-space matrices.
Fifth		Contains the measured actuator position.

## 3D Observer Form $[A(v),B(v),C(v),F(v),H(v)]$

---

Output	Dimension Type	Description
First		Contains the actuator demands.

### Assumptions and Limitations

If the scheduling parameter inputs to the block go out of range, then they are clipped; i.e., the state-space matrices are not interpolated out of range.

### Reference

Hyde, R. A., "H-infinity Aerospace Control Design - A VSTOL Flight Application," Springer Verlag, *Advances in Industrial Control Series*, 1995. ISBN 3-540-19960-8. See Chapter 6.

### See Also

1D Controller  $[A(v),B(v),C(v),D(v)]$

2D Observer Form  $[A(v),B(v),C(v),F(v),H(v)]$

3D Controller  $[A(v),B(v),C(v),D(v)]$

3D Self-Conditioned  $[A(v),B(v),C(v),D(v)]$

# 3D Self-Conditioned [A(v),B(v),C(v),D(v)]

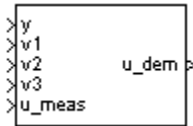
## Purpose

Implement gain-scheduled state-space controller in self-conditioned form depending on two scheduling parameters

## Library

GNC/Control

## Description



The 3D Self-Conditioned [A(v),B(v),C(v),D(v)] block implements a gain-scheduled state-space controller as defined by the equations

$$\dot{x} = A(v)x + B(v)y$$

$$u = C(v)x + D(v)y$$

in the self-conditioned form

$$\dot{z} = (A(v) - H(v)C(v))z + (B(v) - H(v)D(v))e + H(v)u_{meas}$$

$$u_{dem} = C(v)z + D(v)e$$

For the rationale behind this self-conditioned implementation, refer to the Self-Conditioned [A,B,C,D] block reference. These blocks implement a gain-scheduled version of the Self-Conditioned [A,B,C,D] block,  $v$  being the vector of parameters over which  $A$ ,  $B$ ,  $C$ , and  $D$  are defined. This type of controller scheduling assumes that the matrices  $A$ ,  $B$ ,  $C$ , and  $D$  vary smoothly as a function of  $v$ , which is often the case in aerospace applications.

# 3D Self-Conditioned $[A(v), B(v), C(v), D(v)]$

## Dialog Box

Function Block Parameters: 3D Self-Conditioned  $[A(v), B(v), C(v), D(v)]$

StateSpaceSelfCond-3D (mask) (link)

Implement a state-space controller  $[A(v_1, v_2, v_3), B(v_1, v_2, v_3), C(v_1, v_2, v_3), D(v_1, v_2, v_3)]$  in a self-conditioned form. If  $u_{meas} = u_{dem}$ , then the implemented controller is  $[A, B, C, D]$ . If  $u_{meas}$  is limited, e.g., rate limiting, then the poles of the controller become those defined in the mask dialog box. Uses call to Control Systems Toolbox function `place.m` when initializing.  $A, B, C,$  and  $D$  should be 5-D matrices, the last three dimensions corresponding to the scheduling parameters, and the first two corresponding to the matrix for a given set of scheduling parameter values.

Parameters

A-matrix( $v_1, v_2, v_3$ ):  
A

B-matrix( $v_1, v_2, v_3$ ):  
B

C-matrix( $v_1, v_2, v_3$ ):  
C

D-matrix( $v_1, v_2, v_3$ ):  
D

First scheduling variable ( $v_1$ ) breakpoints:  
v1\_vec

Second scheduling variable ( $v_2$ ) breakpoints:  
v2\_vec

Third scheduling variable ( $v_3$ ) breakpoints:  
v3\_vec

Initial state,  $x_{initial}$ :  
0

Poles of  $A(v) - H(v) * C(v) = [w_1 \dots w_n]$ :  
[-5 -2]

OK Cancel Help Apply

### **A-matrix( $\mathbf{v}1, \mathbf{v}2, \mathbf{v}3$ )**

*A*-matrix of the state-space implementation. In the case of 3-D scheduling, the *A*-matrix should have five dimensions, the last three corresponding to scheduling variables  $v1$ ,  $v2$ , and  $v3$ . Hence, for example, if the *A*-matrix corresponding to the first entry of  $v1$ , the first entry of  $v2$ , and the first entry of  $v3$  is the identity matrix, then  $A(:, :, 1, 1, 1) = [1 \ 0; 0 \ 1];$ .

### **B-matrix( $\mathbf{v}1, \mathbf{v}2, \mathbf{v}3$ )**

*B*-matrix of the state-space implementation. In the case of 3-D scheduling, the *B*-matrix should have five dimensions, the last three corresponding to scheduling variables  $v1$ ,  $v2$ , and  $v3$ . Hence, for example, if the *B*-matrix corresponding to the first entry of  $v1$ , the first entry of  $v2$ , and the first entry of  $v3$  is the identity matrix, then  $B(:, :, 1, 1, 1) = [1 \ 0; 0 \ 1];$ .

### **C-matrix( $\mathbf{v}1, \mathbf{v}2, \mathbf{v}3$ )**

*C*-matrix of the state-space implementation. In the case of 3-D scheduling, the *C*-matrix should have five dimensions, the last three corresponding to scheduling variables  $v1$ ,  $v2$ , and  $v3$ . Hence, for example, if the *C*-matrix corresponding to the first entry of  $v1$ , the first entry of  $v2$ , and the first entry of  $v3$  is the identity matrix, then  $C(:, :, 1, 1, 1) = [1 \ 0; 0 \ 1];$ .

### **D-matrix( $\mathbf{v}1, \mathbf{v}2, \mathbf{v}3$ )**

*D*-matrix of the state-space implementation. In the case of 3-D scheduling, the *D*-matrix should have five dimensions, the last three corresponding to scheduling variables  $v1$ ,  $v2$ , and  $v3$ . Hence, for example, if the *D*-matrix corresponding to the first entry of  $v1$ , the first entry of  $v2$ , and the first entry of  $v3$  is the identity matrix, then  $D(:, :, 1, 1, 1) = [1 \ 0; 0 \ 1];$ .

### **First scheduling variable ( $\mathbf{v}1$ ) breakpoints**

Vector of the breakpoints for the first scheduling variable. The length of  $v1$  should be same as the size of the third dimension of *A*, *B*, *C*, and *D*.

## 3D Self-Conditioned $[A(v), B(v), C(v), D(v)]$

### Second scheduling variable ( $v_2$ ) breakpoints

Vector of the breakpoints for the second scheduling variable. The length of  $v_2$  should be same as the size of the fourth dimension of  $A$ ,  $B$ ,  $C$ , and  $D$ .

### Third scheduling variable ( $v_3$ ) breakpoints

Vector of the breakpoints for the third scheduling variable. The length of  $v_3$  should be same as the size of the fifth dimension of  $A$ ,  $B$ ,  $C$ , and  $D$ .

### Initial state, $x_{\text{initial}}$

Vector of initial states for the controller, i.e., initial values for the state vector,  $x$ . It should have length equal to the size of the first dimension of  $A$ .

### Poles of $A(v)-H(v)*C(v)$

Vector of the desired poles of  $A-HC$ . Note that the poles are assigned to the same locations for all values of the scheduling parameter  $v$ . Hence the number of pole locations defined should be equal to the length of the first dimension of the  $A$ -matrix.

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the measurements.
Second		Contains the scheduling variable, ordered conforming to the dimensions of the state-space matrices.
Third		Contains the scheduling variable, ordered conforming to the dimensions of the state-space matrices.
Fourth		Contains the scheduling variable, ordered conforming to the dimensions of the state-space matrices.
Fifth		Contains the measured actuator position.

# 3D Self-Conditioned $[A(v),B(v),C(v),D(v)]$

---

Output	Dimension Type	Description
First		Contains the measured actuator position.

The first input is the measurements.

The second, third, and fourth inputs are the scheduling variables ordered conforming to the dimensions of the state-space matrices.

The fifth input is the measured actuator position.

The output is the actuator demands.

## Assumptions and Limitations

If the scheduling parameter inputs to the block go out of range, then they are clipped; i.e., the state-space matrices are not interpolated out of range.

---

**Note** This block requires the Control System Toolbox product.

---

## Reference

The algorithm used to determine the matrix H is defined in Kautsky, Nichols, and Van Dooren, "Robust Pole Assignment in Linear State Feedback," *International Journal of Control*, Vol. 41, No. 5, pages 1129-1155, 1985.

## See Also

1D Self-Conditioned  $[A(v),B(v),C(v),D(v)]$

2D Self-Conditioned  $[A(v),B(v),C(v),D(v)]$

3D Controller  $[A(v),B(v),C(v),D(v)]$

3D Observer Form  $[A(v),B(v),C(v),F(v),H(v)]$

# 3DoF Animation

---

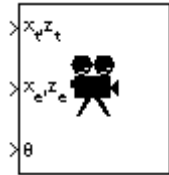
## Purpose

Create 3-D MATLAB Graphics animation of three-degrees-of-freedom object

## Library

Animation

## Description



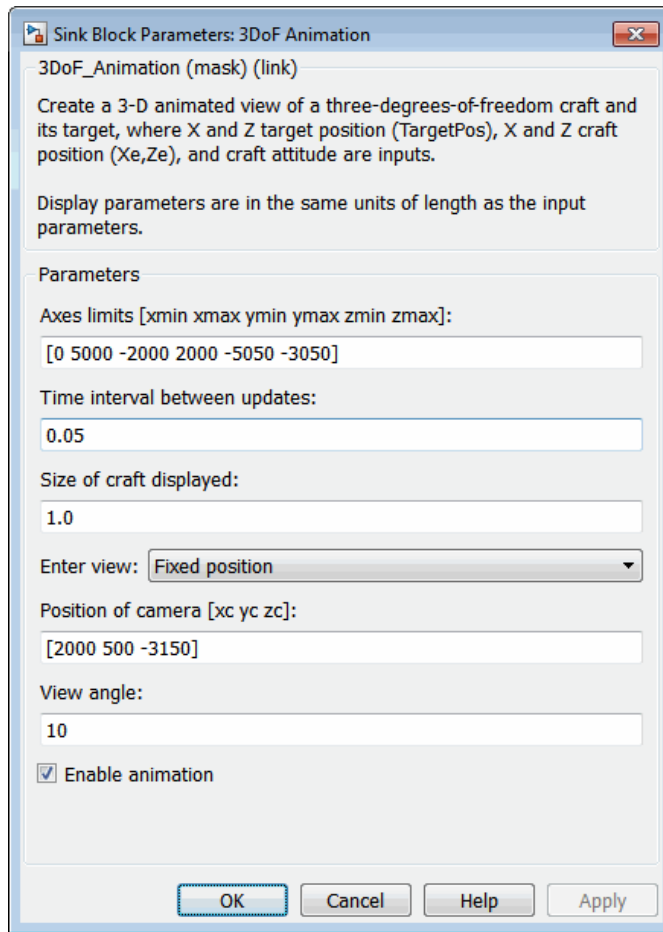
The 3DoF Animation block displays a 3-D animated view of a three-degrees-of-freedom (3DoF) craft, its trajectory, and its target using MATLAB Graphics.

The 3DoF Animation block uses the input values and the dialog parameters to create and display the animation.

This block does not produce deployable code, but can be used with Simulink Coder external mode as a SimViewingDevice.



## Dialog Box



**Axes limits [xmin xmax ymin ymax zmin zmax]**

Specifies the three-dimensional space to be viewed.

**Time interval between updates**

Specifies the time interval at which the animation is redrawn.

**Size of craft displayed**

Scale factor to adjust the size of the craft and target.

# 3DoF Animation

---

## Enter view

Selects preset MATLAB Graphics parameters **CameraTarget** and **CameraUpVector** for the figure axes. The dialog parameters **Position of camera** and **View angle** are used to customize the position and field of view for the selected view. Possible views are

- Fixed position
- Cockpit
- Fly alongside

## Position of camera [xc yc zc]

Specifies the MATLAB Graphics parameter **CameraPosition** for the figure axes. Used in all cases except for the Cockpit view.

## View angle

Specifies the MATLAB Graphics parameter **CameraViewAngle** for the figure axes in degrees.

## Enable animation

When selected, the animation is displayed during the simulation. If not selected, the animation is not displayed.

## Inputs

Input	Dimension Type	Description
First	Vector	Contains the altitude and the downrange position of the target in Earth coordinates.
Second	Vector	Contains the altitude and the downrange position of the craft in Earth coordinates.
Third		Contains the attitude of the craft.

## Examples

See `aero_guidance` for an example of this block.

## See Also

6DoF Animation

FlightGear Preconfigured 6DoF Animation

The figure axes properties `CameraPosition` and `CameraViewAngle`

# 3DoF (Body Axes)

## Purpose

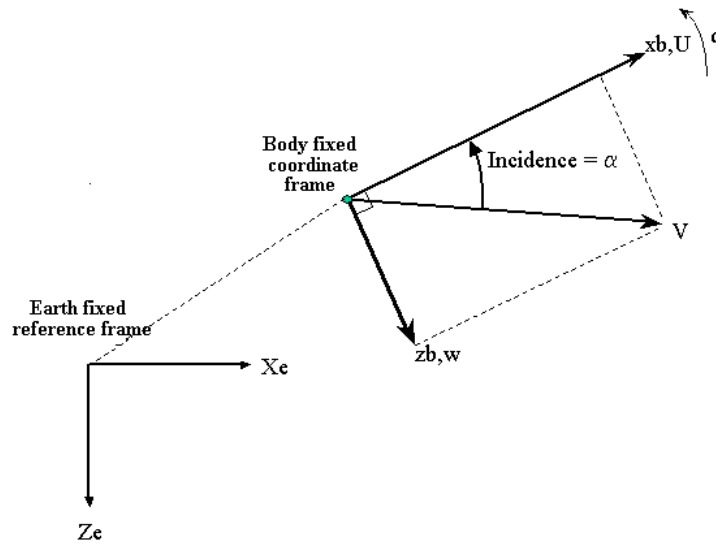
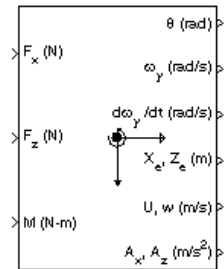
Implement three-degrees-of-freedom equations of motion with respect to body axes

## Library

Equations of Motion/3DoF

## Description

The 3DoF (Body Axes) block considers the rotation in the vertical plane of a body-fixed coordinate frame about an Earth-fixed reference frame.



The equations of motion are

$$\dot{u} = \frac{F_x}{m} - qw - g \sin \theta$$

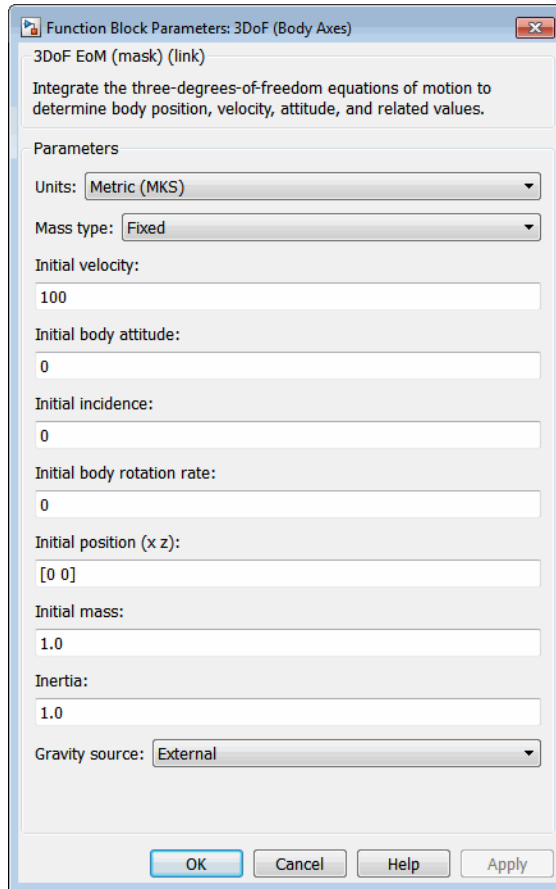
$$\dot{w} = \frac{F_z}{m} + qu + g \cos \theta$$

$$\dot{q} = \frac{M}{I_{yy}}$$

$$\dot{\theta} = q$$

where the applied forces are assumed to act at the center of gravity of the body.

## Dialog Box



The dialog box is titled "Function Block Parameters: 3DoF (Body Axes)". It contains the following fields and options:

- 3DoF EoM (mask) (link)
- Integrate the three-degrees-of-freedom equations of motion to determine body position, velocity, attitude, and related values.
- Parameters section:
  - Units: Metric (MKS)
  - Mass type: Fixed
  - Initial velocity: 100
  - Initial body attitude: 0
  - Initial incidence: 0
  - Initial body rotation rate: 0
  - Initial position (x z): [0 0]
  - Initial mass: 1.0
  - Inertia: 1.0
  - Gravity source: External
- Buttons: OK, Cancel, Help, Apply

### Units

Specifies the input and output units:

# 3DoF (Body Axes)

---

Units	Forces	Moment	Acceleration	Velocity	Position	Mass	Inertia
Metric (MKS)	Newton	Newton-meter	Meters per second squared	Meters per second	Meters	Kilogram	Kilogram meter squared
English (Velocity in ft/s)	Pound	Foot-pound	Feet per second squared	Feet per second	Feet	Slug	Slug foot squared
English (Velocity in kts)	Pound	Foot-pound	Feet per second squared	Knots	Feet	Slug	Slug foot squared

## Mass Type

Select the type of mass to use:

Fixed	Mass is constant throughout the simulation.
Simple Variable	Mass and inertia vary linearly as a function of mass rate.
Custom Variable	Mass and inertia variations are customizable.

The Fixed selection conforms to the previously described equations of motion.

## Initial velocity

A scalar value for the initial velocity of the body, ( $V_0$ ).

## Initial body attitude

A scalar value for the initial pitch attitude of the body, ( $\theta_0$ ).

## Initial incidence

A scalar value for the initial angle between the velocity vector and the body, ( $\alpha_0$ ).

## Initial body rotation rate

A scalar value for the initial body rotation rate, ( $q_0$ ).

## Initial position (x,z)

A two-element vector containing the initial location of the body in the Earth-fixed reference frame.

## Initial Mass

A scalar value for the mass of the body.

## Inertia

A scalar value for the inertia of the body.

## Gravity Source

Specify source of gravity:

External	Variable gravity input to block
Internal	Constant gravity specified in mask

## Acceleration due to gravity

A scalar value for the acceleration due to gravity used if internal gravity source is selected. If gravity is to be neglected in the simulation, this value can be set to 0.

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the force acting along the body $x$ -axis, ( $F_x$ ).
Second		Contains the force acting along the body $z$ -axis, ( $F_z$ ).
Third		Contains the applied pitch moment, ( $M$ ).
Fourth (Optional)		Contains the block is gravity in the selected units.

## 3DoF (Body Axes)

---

Output	Dimension Type	Description
First		Contains the pitch attitude, in radians ( $\theta$ ).
Second		Contains the pitch angular rate, in radians per second ( $\dot{q}$ ).
Third		Contains the pitch angular acceleration, in radians per second squared ( $\ddot{q}$ ).
Fourth	Two-element vector	Contains the location of the body, in the Earth-fixed reference frame, ( $X_e, Z_e$ ).
Fifth	Two-element vector	Contains the velocity of the body resolved into the body-fixed coordinate frame, ( $u, w$ ).
Sixth	Two-element vector	Contains the acceleration of the body resolved into the body-fixed coordinate frame, ( $A_x, A_z$ ).

### Examples

See `aero_guidance` for an example of this block.

### See Also

3DoF (Wind Axes)

4th Order Point Mass (Longitudinal)

Custom Variable Mass 3DoF (Body Axes)

Custom Variable Mass 3DoF (Wind Axes)

Simple Variable Mass 3DoF (Body Axes)

Simple Variable Mass 3DoF (Wind Axes)



## Purpose

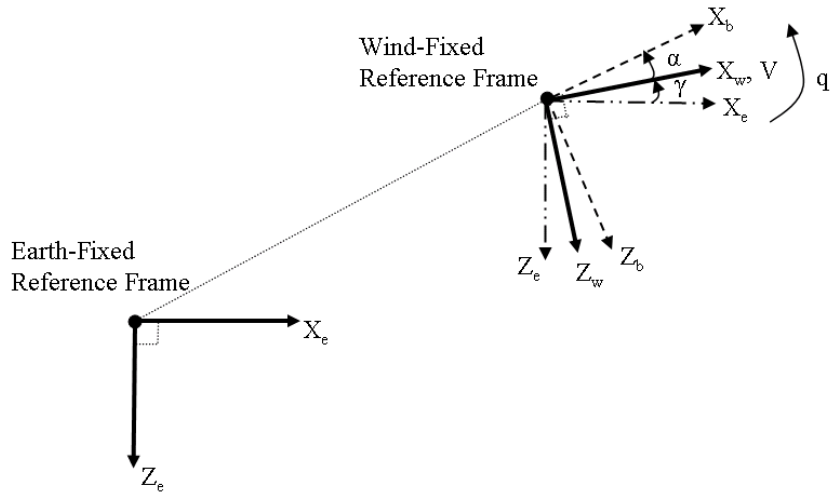
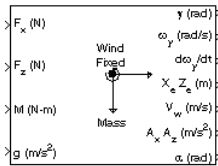
Implement three-degrees-of-freedom equations of motion with respect to wind axes

## Library

Equations of Motion/3DoF

## Description

The 3DoF (Wind Axes) block considers the rotation in the vertical plane of a wind-fixed coordinate frame about an Earth-fixed reference frame.



The equations of motion are

$$\dot{V} = \frac{F_{x_{wind}}}{m} - g \sin \gamma$$

$$\dot{\alpha} = \frac{F_{z_{wind}}}{mV \cos \beta} + q + \frac{g}{V \cos \beta} \cos \gamma$$

$$\dot{q} = \dot{\theta} = \frac{M_{y_{body}}}{I_{yy}}$$

$$\dot{\gamma} = q - \dot{\alpha}$$

# 3DoF (Wind Axes)

where the applied forces are assumed to act at the center of gravity of the body.

## Dialog Box

Function Block Parameters: 3DoF (Wind Axes)

3DoF Wind EoM (mask) (link)

Integrate the three-degrees-of-freedom equations of motion in wind axes to determine position, velocity, attitude, and related values.

Parameters

Units: Metric (MKS)

Mass type: Fixed

Initial airspeed: 100

Initial flight path angle: 0

Initial incidence: 0

Initial body rotation rate: 0

Initial position (x z): [0 0]

Initial mass: 1.0

Inertia body axes: 1.0

Gravity source: External

OK Cancel Help Apply

## Units

Specifies the input and output units:

Units	Forces	Moment	Acceleration	Velocity	Position	Mass	Inertia
Metric (MKS)	Newton	Newton meter	Meters per second squared	Meters per second	Meters	Kilogram	Kilogram meter squared
English (Velocity in ft/s)	Pound	Foot pound	Feet per second squared	Feet per second	Feet	Slug	Slug foot squared
English (Velocity in kts)	Pound	Foot pound	Feet per second squared	Knots	Feet	Slug	Slug foot squared

## Mass Type

Select the type of mass to use:

Fixed	Mass is constant throughout the simulation.
Simple Variable	Mass and inertia vary linearly as a function of mass rate.
Custom Variable	Mass and inertia variations are customizable.

The Fixed selection conforms to the previously described equations of motion.

## Initial airspeed

A scalar value for the initial velocity of the body, ( $V_0$ ).

## Initial flight path angle

A scalar value for the initial flight path angle of the body, ( $\gamma_0$ ).

## Initial incidence

A scalar value for the initial angle between the velocity vector and the body, ( $\alpha_0$ ).

## 3DoF (Wind Axes)

---

### Initial body rotation rate

A scalar value for the initial body rotation rate, ( $q_0$ ).

### Initial position (x,z)

A two-element vector containing the initial location of the body in the Earth-fixed reference frame.

### Initial Mass

A scalar value for the mass of the body.

### Inertia body axes

A scalar value for the inertia of the body.

### Gravity Source

Specify source of gravity:

External	Variable gravity input to block
Internal	Constant gravity specified in mask

### Acceleration due to gravity

A scalar value for the acceleration due to gravity used if internal gravity source is selected. If gravity is to be neglected in the simulation, this value can be set to 0.

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the force acting along the wind $x$ -axis, ( $F_x$ ).
Second		Contains the force acting along the wind $z$ -axis, ( $F_z$ ).
Third		Contains the applied pitch moment in body axes, ( $M$ ).
Fourth (Optional)		Contains the block is gravity in the selected units.

Output	Dimension Type	Description
First		Contains the flight path angle, in radians ( $\gamma$ ).
Second		Contains the pitch angular rate, in radians per second ( $\omega_y$ ).
Third		Contains the pitch angular acceleration, in radians per second squared ( $d\omega_y/dt$ ).
Fourth	Two-element vector	Contains the location of the body, in the Earth-fixed reference frame, ( $X_e, Z_e$ ).
Fifth	Two-element vector	Contains the velocity of the body resolved into the wind-fixed coordinate frame, ( $V, 0$ ).
Sixth	Two-element vector	Contains the acceleration of the body resolved into the body-fixed coordinate frame, ( $A_x, A_z$ ).
Seventh	Scalar	Contains the angle of attack, ( $\alpha$ ).

### Assumptions and Limitations

The block assumes that the applied forces are acting at the center of gravity of the body, and that the mass and inertia are constant.

### Reference

Stevens, B. L., and F. L. Lewis, *Aircraft Control and Simulation*, John Wiley & Sons, New York, 1992.

### See Also

3DoF (Body Axes)

4th Order Point Mass (Longitudinal)

Custom Variable Mass 3DoF (Body Axes)

Custom Variable Mass 3DoF (Wind Axes)

## 3DoF (Wind Axes)

---

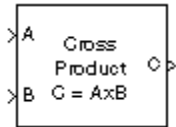
Simple Variable Mass 3DoF (Body Axes)

Simple Variable Mass 3DoF (Wind Axes)

**Purpose** Calculate cross product of two 3-by-1 vectors

**Library** Utilities/Math Operations

**Description**



The 3x3 Cross Product block computes cross (or vector) product of two vectors,  $A$  and  $B$ , by generating a third vector,  $C$ , in a direction normal to the plane containing  $A$  and  $B$ , and with magnitude equal to the product of the lengths of  $A$  and  $B$  multiplied by the sine of the angle between them. The direction of  $C$  is that in which a right-handed screw would move in turning from  $A$  to  $B$ .

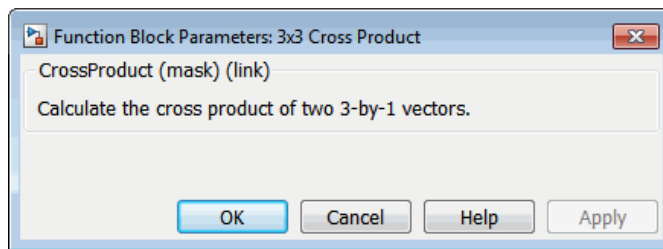
$$A = a_1\mathbf{i} + a_2\mathbf{j} + a_3\mathbf{k}$$

$$B = b_1\mathbf{i} + b_2\mathbf{j} + b_3\mathbf{k}$$

$$C = A \times B = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix}$$

$$= (a_2b_3 - a_3b_2)\mathbf{i} + (a_3b_1 - a_1b_3)\mathbf{j} + (a_1b_2 - a_2b_1)\mathbf{k}$$

**Dialog Box**



**Inputs and Outputs**

Input	Dimension Type	Description
First	3-by-1 vector	
Second	3-by-1 vector	

## 3x3 Cross Product

---

Output	Dimension Type	Description
First	3-by-1 vector	



# 4th Order Point Mass (Longitudinal)

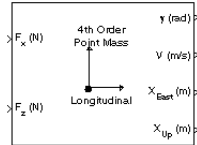
## Purpose

Calculate fourth-order point mass

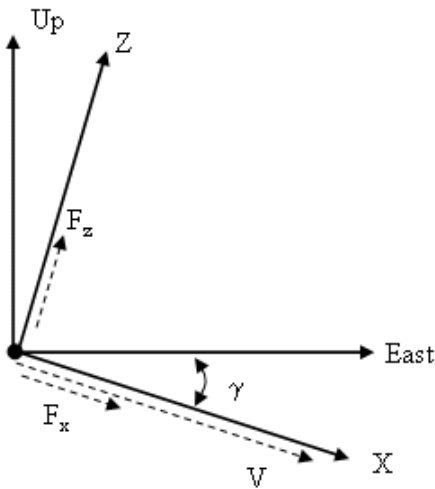
## Library

Equations of Motion/Point Mass

## Description



The 4th Order Point Mass (Longitudinal) block performs the calculations for the translational motion of a single point mass or multiple point masses.

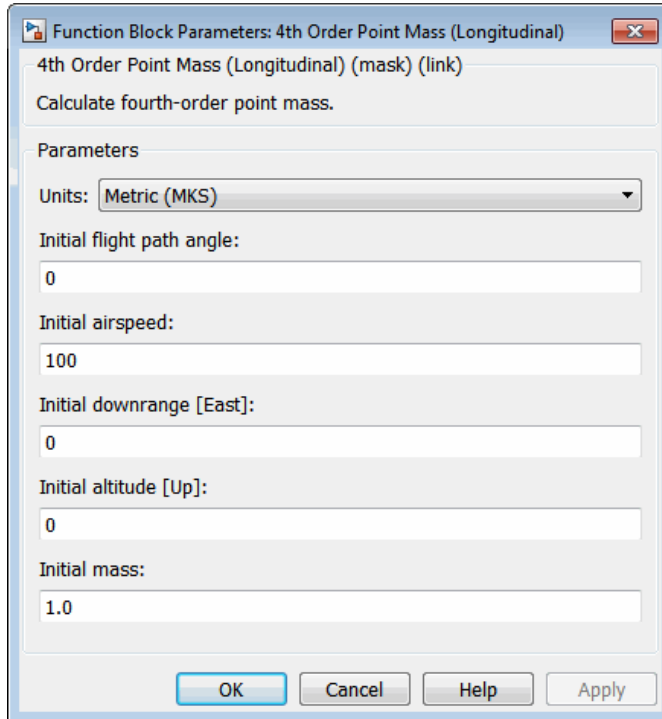


The translational motions of the point mass  $[X_{East} \ X_{Up}]^T$  are functions of airspeed ( $V$ ) and flight path angle ( $\gamma$ ),

$$\begin{aligned}F_x &= m\dot{V} \\F_z &= mV\dot{\gamma} \\ \dot{X}_{East} &= V \cos \gamma \\ \dot{X}_{Up} &= V \sin \gamma\end{aligned}$$

# 4th Order Point Mass (Longitudinal)

where the applied forces  $[F_x, F_z]^T$  are in a system defined as follows:  $x$ -axis is in the direction of vehicle velocity relative to air,  $z$ -axis is upward, and  $y$ -axis completes the right-handed frame. The mass of the body  $m$  is assumed constant.



## Dialog Box

### Units

Specifies the input and output units:

# 4th Order Point Mass (Longitudinal)

---

<b>Units</b>	<b>Forces</b>	<b>Velocity</b>	<b>Position</b>
Metric (MKS)	Newton	Meters per second	Meters
English (Velocity in ft/s)	Pound	Feet per second	Feet
English (Velocity in kts)	Pound	Knots	Feet

## **Initial flight path angle**

The scalar or vector containing the initial flight path angle of the point mass(es).

## **Initial airspeed**

The scalar or vector containing the initial airspeed of the point mass(es).

## **Initial downrange**

The scalar or vector containing the initial downrange of the point mass(es).

## **Initial altitude**

The scalar or vector containing the initial altitude of the point mass(es).

## **Initial mass**

The scalar or vector containing the mass of the point mass(es).

## **Inputs and Outputs**

<b>Input</b>	<b>Dimension Type</b>	<b>Description</b>
First		Contains the force in $x$ -axis in selected units.
Second		Contains the force in $z$ -axis in selected units.

# 4th Order Point Mass (Longitudinal)

---

Output Dimension Type	Description
First	Contains the flight path angle in radians.
Second	Contains the airspeed in selected units.
Third	Contains the downrange or amount traveled East in selected units.
Fourth	Contains the altitude or amount traveled Up in selected units.

## Assumptions and Limitations

The flat Earth reference frame is considered inertial, an excellent approximation that allows the forces due to the Earth's motion relative to the "fixed stars" to be neglected.

## See Also

4th Order Point Mass Forces (Longitudinal)  
3DoF (Body Axes)  
3DoF (Wind Axes)  
6th Order Point Mass (Coordinated Flight)  
6th Order Point Mass Forces (Coordinated Flight)  
Custom Variable Mass 3DoF (Body Axes)  
Custom Variable Mass 3DoF (Wind Axes)  
Simple Variable Mass 3DoF (Body Axes)  
Simple Variable Mass 3DoF (Wind Axes)

# 4th Order Point Mass Forces (Longitudinal)

## Purpose

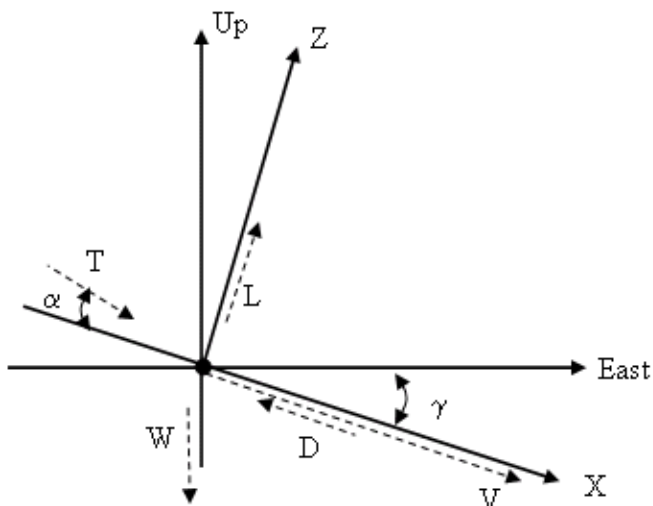
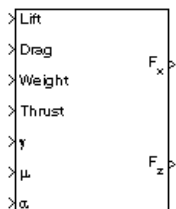
Calculate forces used by fourth-order point mass

## Library

Equations of Motion/Point Mass

## Description

The 4th Order Point Mass Forces (Longitudinal) block calculates the applied forces for a single point mass or multiple point masses.



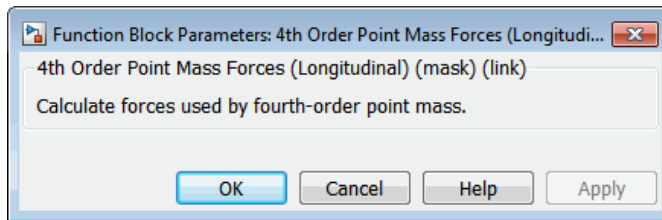
The applied forces  $[F_x \ F_z]^T$  are in a system defined as follows:  $x$ -axis is in the direction of vehicle velocity relative to air,  $z$ -axis is upward, and  $y$ -axis completes the right-handed frame. They are functions of lift ( $L$ ), drag ( $D$ ), thrust ( $T$ ), weight ( $W$ ), flight path angle ( $\gamma$ ), angle of attack ( $\alpha$ ), and bank angle ( $\mu$ ).

$$F_z = (L + T \sin \alpha) \cos \mu - W \cos \gamma$$

$$F_x = T \cos \alpha - D - W \sin \gamma$$

# 4th Order Point Mass Forces (Longitudinal)

## Dialog Box



## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the lift in units of force.
Second		Contains the drag in units of force.
Third		Contains the weight in units of force.
Fourth		Contains the thrust in units of force.
Fifth		Contains the flight path angle in radians.
Sixth		Contains the bank angle in radians.
Seventh		Contains the angle of attack in radians.

Output	Dimension Type	Description
First		Contains the force in $x$ -axis in units of force.
Second		Contains the force in $z$ -axis in units of force.

## Assumptions and Limitations

The flat Earth reference frame is considered inertial, an excellent approximation that allows the forces due to the Earth's motion relative to the "fixed stars" to be neglected.

## See Also

4th Order Point Mass (Longitudinal)

6th Order Point Mass (Coordinated Flight)

# 4th Order Point Mass Forces (Longitudinal)

---

6th Order Point Mass Forces (Coordinated Flight)

# 6DoF Animation

---

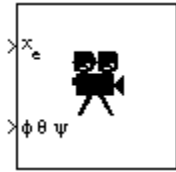
## Purpose

Create 3-D MATLAB Graphics animation of six-degrees-of-freedom object

## Library

Animation

## Description



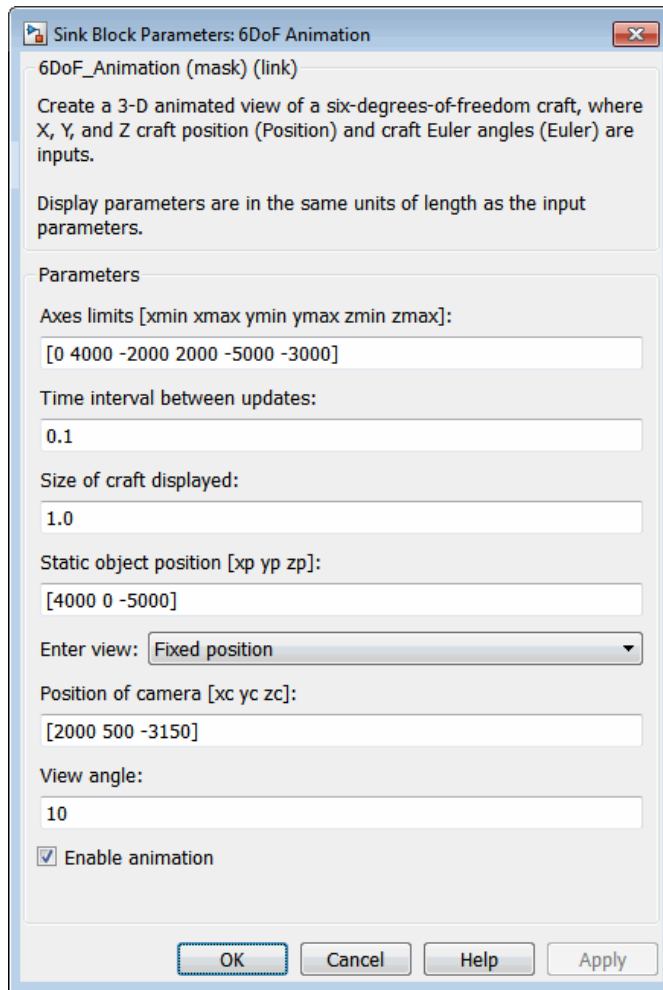
The 6DoF Animation block displays a 3-D animated view of a six-degrees-of-freedom (6DoF) craft, its trajectory, and its target using MATLAB Graphics.

The 6DoF Animation block uses the input values and the dialog parameters to create and display the animation.

This block does not produce deployable code, but can be used with Simulink Coder external mode as a SimViewingDevice.



## Dialog Box



**Axes limits [xmin xmax ymin ymax zmin zmax]**  
Specifies the three-dimensional space to be viewed.

**Time interval between updates**  
Specifies the time interval at which the animation is redrawn.

# 6DoF Animation

---

## Size of craft displayed

Scale factor to adjust the size of the craft and target.

## Static object position

Specifies the altitude, the crossrange position, and the downrange position of the target.

## Enter view

Selects preset MATLAB Graphics parameters **CameraTarget** and **CameraUpVector** for the figure axes. The dialog parameters **Position of camera** and **View angle** are used to customize the position and field of view for the selected view. Possible views are

- Fixed position
- Cockpit
- Fly alongside

## Position of camera [xc yc zc]

Specifies the MATLAB Graphics parameter **CameraPosition** for the figure axes. Used in all cases except for the Cockpit view.

## View angle

Specifies the MATLAB Graphics parameter **CameraViewAngle** for the figure axes in degrees.

## Enable animation

When selected, the animation is displayed during the simulation. If not selected, the animation is not displayed.

## Inputs

Input	Dimension Type	Description
First	Vector	Contains the altitude, the crossrange position, and the downrange position of the craft in Earth coordinates.
Second	Vector	Contains the Euler angles of the craft.

### **See Also**

3DoF Animation

FlightGear Preconfigured 6DoF Animation

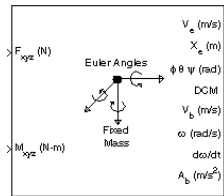
The figure axes properties `CameraPosition` and `CameraViewAngle`

# 6DoF (Euler Angles)

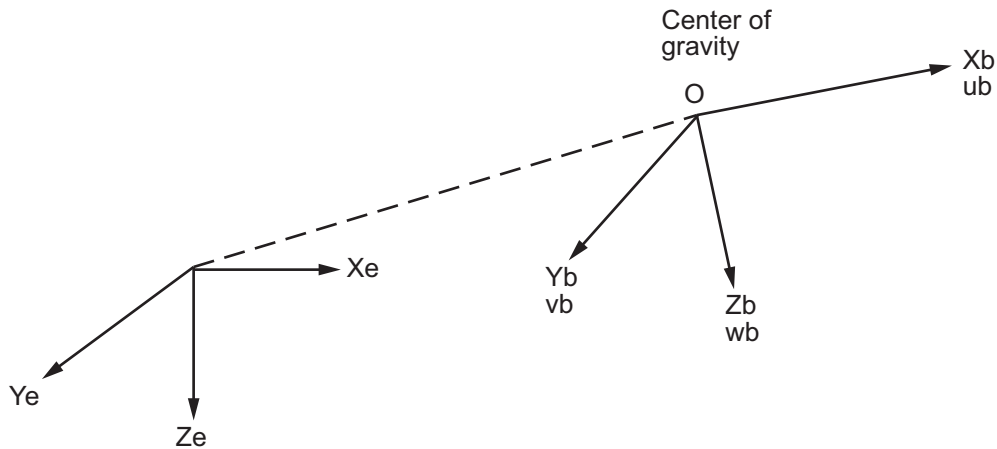
**Purpose** Implement Euler angle representation of six-degrees-of-freedom equations of motion

**Library** Equations of Motion/6DoF

## Description



The 6DoF (Euler Angles) block considers the rotation of a body-fixed coordinate frame ( $X_b, Y_b, Z_b$ ) about a flat Earth reference frame ( $X_e, Y_e, Z_e$ ). The origin of the body-fixed coordinate frame is the center of gravity of the body, and the body is assumed to be rigid, an assumption that eliminates the need to consider the forces acting between individual elements of mass. The flat Earth reference frame is considered inertial, an excellent approximation that allows the forces due to the Earth's motion relative to the "fixed stars" to be neglected.



Flat Earth reference frame

The translational motion of the body-fixed coordinate frame is given below, where the applied forces  $[F_x, F_y, F_z]^T$  are in the body-fixed frame, and the mass of the body  $m$  is assumed constant.

$$\bar{\mathbf{F}}_b = \begin{bmatrix} \mathbf{F}_x \\ \mathbf{F}_y \\ \mathbf{F}_z \end{bmatrix} = m \left( \dot{\bar{\mathbf{V}}}_b + \bar{\boldsymbol{\omega}} \times \bar{\mathbf{V}}_b \right)$$

$$\bar{\mathbf{V}}_b = \begin{bmatrix} u_b \\ v_b \\ w_b \end{bmatrix}, \bar{\boldsymbol{\omega}} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

The rotational dynamics of the body-fixed frame are given below, where the applied moments are  $[L \ M \ N]^T$ , and the inertia tensor  $I$  is with respect to the origin O.

$$\bar{\mathbf{M}}_B = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = I \dot{\bar{\boldsymbol{\omega}}} + \bar{\boldsymbol{\omega}} \times (I \bar{\boldsymbol{\omega}})$$

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix}$$

The relationship between the body-fixed angular velocity vector,  $[p \ q \ r]^T$ , and the rate of change of the Euler angles,  $[\dot{\phi} \ \dot{\theta} \ \dot{\psi}]^T$ , can be determined by resolving the Euler rates into the body-fixed coordinate frame.

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & \sin \phi \\ 0 & -\sin \phi & \cos \phi \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & \sin \phi \\ 0 & -\sin \phi & \cos \phi \end{bmatrix} \begin{bmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ \sin \theta & 0 & \cos \theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \equiv \mathbf{J}^{-1} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

# 6DoF (Euler Angles)

Inverting  $J$  then gives the required relationship to determine the Euler rate vector.

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = J \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & (\sin \phi \tan \theta) & (\cos \phi \tan \theta) \\ 0 & \cos \phi & -\sin \phi \\ 0 & \frac{\sin \phi}{\cos \theta} & \frac{\cos \phi}{\cos \theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

## Dialog Box

Function Block Parameters: 6DoF (Euler Angles)

6DoF EoM (Body Axis) (mask) (link)

Integrate the six-degrees-of-freedom equations of motion in body axis.

Parameters

Units: Metric (MKS)

Mass type: Fixed

Representation: Euler Angles

Initial position in inertial axes [Xe,Ye,Ze]:  
[ 0 0 0 ]

Initial velocity in body axes [U,v,w]:  
[ 0 0 0 ]

Initial Euler orientation [roll, pitch, yaw]:  
[ 0 0 0 ]

Initial body rotation rates [p,q,r]:  
[ 0 0 0 ]

Initial mass:  
1.0

Inertia:  
eye(3)

OK Cancel Help Apply

## Units

Specifies the input and output units:

<b>Units</b>	<b>Forces</b>	<b>Moment</b>	<b>Acceleration</b>	<b>Velocity</b>	<b>Position</b>	<b>Mass</b>	<b>Inertia</b>
Metric (MKS)	Newton	Newton meter	Meters per second squared	Meters per second	Meters	Kilogram	Kilogram meter squared
English (Velocity in ft/s)	Pound	Foot pound	Feet per second squared	Feet per second	Feet	Slug	Slug foot squared
English (Velocity in kts)	Pound	Foot pound	Feet per second squared	Knots	Feet	Slug	Slug foot squared

## Mass Type

Select the type of mass to use:

Fixed	Mass is constant throughout the simulation.
Simple Variable	Mass and inertia vary linearly as a function of mass rate.
Custom Variable	Mass and inertia variations are customizable.

The Fixed selection conforms to the previously described equations of motion.

## Representation

Select the representation to use:

Euler Angles	Use Euler angles within equations of motion.
Quaternion	Use quaternions within equations of motion.

# 6DoF (Euler Angles)

---

The Euler Angles selection conforms to the previously described equations of motion.

## **Initial position in inertial axes**

The three-element vector for the initial location of the body in the flat Earth reference frame.

## **Initial velocity in body axes**

The three-element vector for the initial velocity in the body-fixed coordinate frame.

## **Initial Euler rotation**

The three-element vector for the initial Euler rotation angles [roll, pitch, yaw], in radians.

## **Initial body rotation rates**

The three-element vector for the initial body-fixed angular rates, in radians per second.

## **Initial Mass**

The mass of the rigid body.

## **Inertia**

The 3-by-3 inertia tensor matrix  $I$ .

## **Inputs and Outputs**

<b>Input</b>	<b>Dimension Type</b>	<b>Description</b>
First	Vector	Contains the three applied forces in body-fixed coordinate frame.
Second	Vector	Contains the three applied moments in body-fixed coordinate frame.

<b>Output</b>	<b>Dimension Type</b>	<b>Description</b>
First	Three-element vector	Contains the velocity in the flat Earth reference frame.
Second	Three-element vector	Contains the position in the flat Earth reference frame.



Output	Dimension Type	Description
Third	Three-element vector	Contains the Euler rotation angles [roll, pitch, yaw], in radians.
Fourth	3-by-3 matrix	Contains the coordinate transformation from flat Earth axes to body-fixed axes.
Fifth	Three-element vector	Contains the velocity in the body-fixed frame.
Sixth	Three-element vector	Contains the angular rates in body-fixed axes, in radians per second.
Seventh	Three-element vector	Contains the angular accelerations in body-fixed axes, in radians per second squared.
Eighth	Three-element vector	Contains the accelerations in body-fixed axes.

### Assumptions and Limitations

The block assumes that the applied forces are acting at the center of gravity of the body, and that the mass and inertia are constant.

### Examples

See the `aeroblk_six_dof` airframe in `aeroblk_HL20` and `asbh120` for examples of this block.

### Reference

Mangiacasale, L., *Flight Mechanics of a  $\mu$ -Airplane with a MATLAB Simulink Helper*, Edizioni Libreria CLUP, Milan, 1998.

### See Also

6DoF (Quaternion)  
6DoF ECEF (Quaternion)  
6DoF Wind (Quaternion)  
6DoF Wind (Wind Angles)

## 6DoF (Euler Angles)

---

6th Order Point Mass (Coordinated Flight)  
Custom Variable Mass 6DoF (Euler Angles)  
Custom Variable Mass 6DoF (Quaternion)  
Custom Variable Mass 6DoF ECEF (Quaternion)  
Custom Variable Mass 6DoF Wind (Quaternion)  
Custom Variable Mass 6DoF Wind (Wind Angles)  
Simple Variable Mass 6DoF (Euler Angles)  
Simple Variable Mass 6DoF (Quaternion)  
Simple Variable Mass 6DoF ECEF (Quaternion)  
Simple Variable Mass 6DoF Wind (Quaternion)  
Simple Variable Mass 6DoF Wind (Wind Angles)

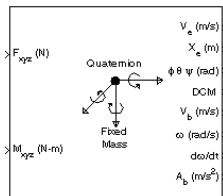
## Purpose

Implement quaternion representation of six-degrees-of-freedom equations of motion with respect to body axes

## Library

Equations of Motion/6DoF

## Description



For a description of the coordinate system and the translational dynamics, see the block description for the 6DoF (Euler Angles) block.

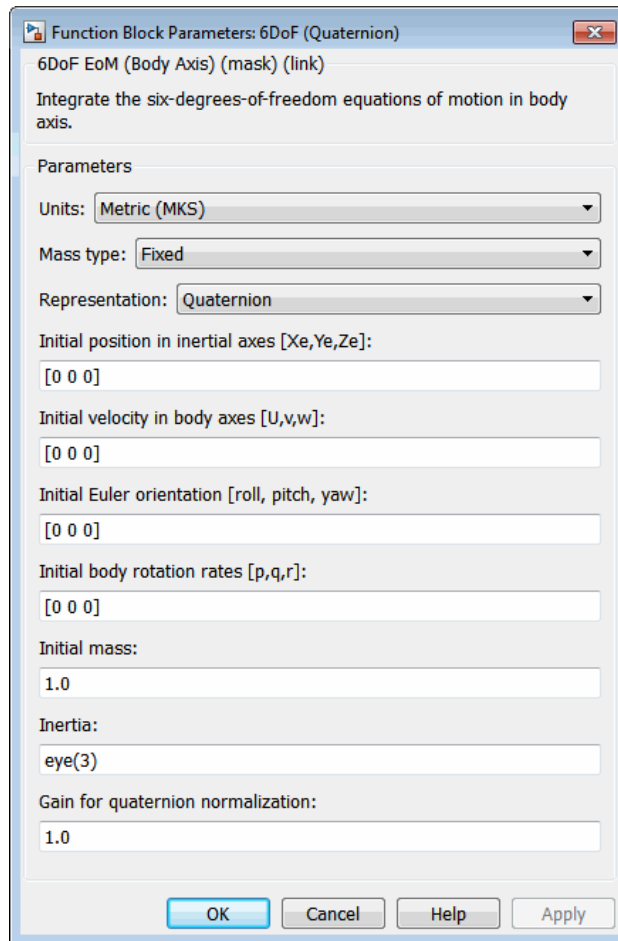
The integration of the rate of change of the quaternion vector is given below. The gain  $K$  drives the norm of the quaternion state vector to 1.0 should  $\varepsilon$  become nonzero. You must choose the value of this gain with care, because a large value improves the decay rate of the error in the norm, but also slows the simulation because fast dynamics are introduced. An error in the magnitude in one element of the quaternion vector is spread equally among all the elements, potentially increasing the error in the state vector.

$$\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} + K \varepsilon \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

$$\varepsilon = 1 - (q_0^2 + q_1^2 + q_2^2 + q_3^2)$$

# 6DoF (Quaternion)

## Dialog Box



### Units

Specifies the input and output units:

<b>Units</b>	<b>Forces</b>	<b>Moment</b>	<b>Acceleration</b>	<b>Velocity</b>	<b>Position</b>	<b>Mass</b>	<b>Inertia</b>
Metric (MKS)	Newton	Newton meter	Meters per second squared	Meters per second	Meters	Kilogram	Kilogram meter squared
English (Velocity in ft/s)	Pound	Foot pound	Feet per second squared	Feet per second	Feet	Slug	Slug foot squared
English (Velocity in kts)	Pound	Foot pound	Feet per second squared	Knots	Feet	Slug	Slug foot squared

## Mass Type

Select the type of mass to use:

- Fixed                      Mass is constant throughout the simulation.
- Simple Variable        Mass and inertia vary linearly as a function of mass rate.
- Custom Variable        Mass and inertia variations are customizable.

The Fixed selection conforms to the previously described equations of motion.

## Representation

Select the representation to use:

- Euler Angles            Use Euler angles within equations of motion.
- Quaternion              Use quaternions within equations of motion.

# 6DoF (Quaternion)

---

The Quaternion selection conforms to the previously described equations of motion.

## **Initial position in inertial axes**

The three-element vector for the initial location of the body in the flat Earth reference frame.

## **Initial velocity in body axes**

The three-element vector for the initial velocity in the body-fixed coordinate frame.

## **Initial Euler rotation**

The three-element vector for the initial Euler rotation angles [roll, pitch, yaw], in radians.

## **Initial body rotation rates**

The three-element vector for the initial body-fixed angular rates, in radians per second.

## **Initial Mass**

The mass of the rigid body.

## **Inertia matrix**

The 3-by-3 inertia tensor matrix  $I$ .

## **Gain for quaternion normalization**

The gain to maintain the norm of the quaternion vector equal to 1.0.

## **Inputs and Outputs**

<b>Input</b>	<b>Dimension Type</b>	<b>Description</b>
First	Vector	Contains the three applied forces.
Second	Vector	Contains the three applied moments.

Output	Dimension Type	Description
First	Three-element vector	Contains the velocity in the flat Earth reference frame.
Second	Three-element vector	Contains the position in the flat Earth reference frame.
Third	Three-element vector	Contains the Euler rotation angles [roll, pitch, yaw], in radians.
Fourth	3-by-3 matrix	Contains the coordinate transformation from flat Earth axes to body-fixed axes.
Fifth	Three-element vector	Contains the velocity in the body-fixed frame.
Sixth	Three-element vector	Contains the angular rates in body-fixed axes, in radians per second.
Seventh	Three-element vector	Contains the angular accelerations in body-fixed axes, in radians per second squared.
Eight	Three-element vector	Contains the accelerations in body-fixed axes.

### Assumptions and Limitations

The block assumes that the applied forces are acting at the center of gravity of the body, and that the mass and inertia are constant.

### Reference

Mangiaccasale, L., *Flight Mechanics of a  $\mu$ -Airplane with a MATLAB Simulink Helper*, Edizioni Libreria CLUP, Milan, 1998.

### See Also

6DoF (Euler Angles)  
6DoF ECEF (Quaternion)  
6DoF Wind (Quaternion)

## 6DoF (Quaternion)

---

6DoF Wind (Wind Angles)  
6th Order Point Mass (Coordinated Flight)  
Custom Variable Mass 6DoF (Euler Angles)  
Custom Variable Mass 6DoF (Quaternion)  
Custom Variable Mass 6DoF ECEF (Quaternion)  
Custom Variable Mass 6DoF Wind (Quaternion)  
Custom Variable Mass 6DoF Wind (Wind Angles)  
Simple Variable Mass 6DoF (Euler Angles)  
Simple Variable Mass 6DoF (Quaternion)  
Simple Variable Mass 6DoF ECEF (Quaternion)  
Simple Variable Mass 6DoF Wind (Quaternion)  
Simple Variable Mass 6DoF Wind (Wind Angles)



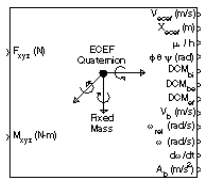
## Purpose

Implement quaternion representation of six-degrees-of-freedom equations of motion in Earth-centered Earth-fixed (ECEF) coordinates

## Library

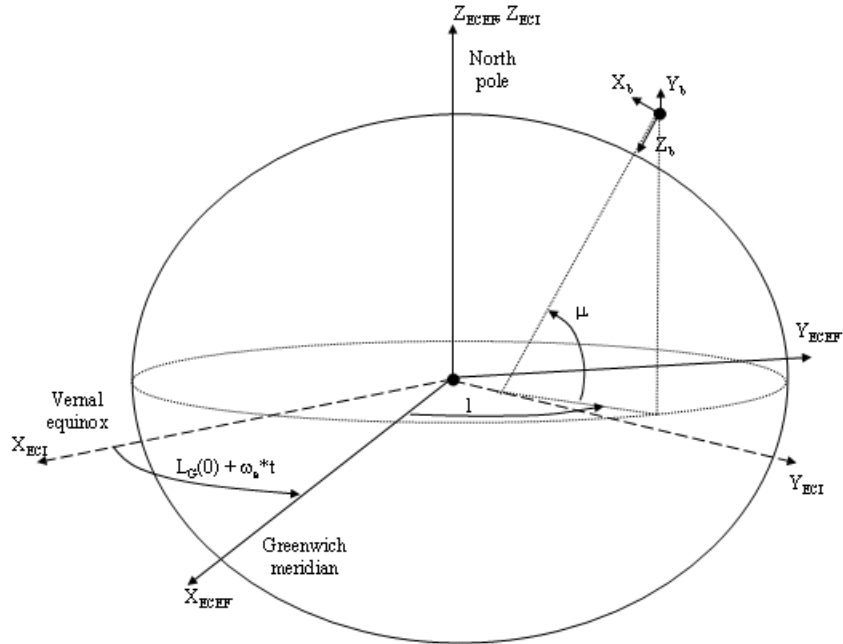
Equations of Motion/6DoF

## Description



The 6DoF ECEF (Quaternion) block considers the rotation of a Earth-centered Earth-fixed (ECEF) coordinate frame ( $X_{ECEF}$ ,  $Y_{ECEF}$ ,  $Z_{ECEF}$ ) about an Earth-centered inertial (ECI) reference frame ( $X_{ECI}$ ,  $Y_{ECI}$ ,  $Z_{ECI}$ ). The origin of the ECEF coordinate frame is the center of the Earth, additionally the body of interest is assumed to be rigid, an assumption that eliminates the need to consider the forces acting between individual elements of mass. The representation of the rotation of ECEF frame from ECI frame is simplified to consider only the constant rotation of the ellipsoid Earth ( $\omega_e$ ) including an initial celestial longitude ( $L_G(0)$ ). This excellent approximation allows the forces due to the Earth's complex motion relative to the "fixed stars" to be neglected.

# 6DoF ECEF (Quaternion)



The translational motion of the ECEF coordinate frame is given below, where the applied forces  $[F_x \ F_y \ F_z]^T$  are in the body frame, and the mass of the body  $m$  is assumed constant.

$$\bar{F}_b = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = m(\dot{\bar{V}}_b + \bar{\omega}_b \times \bar{V}_b + DCM_{bf} \bar{\omega}_e \times \bar{V}_b) + DCM_{bf}(\bar{\omega}_e \times (\bar{\omega}_e \times \bar{X}_f))$$

where the change of position in ECEF  $\dot{\bar{x}}_f$  is calculated by

$$\dot{\bar{x}}_f = DCM_{fb} \bar{V}_b$$

and the velocity of the body with respect to ECEF frame, expressed in body frame ( $\bar{V}_b$ ), angular rates of the body with respect to ECI frame,

expressed in body frame ( $\bar{\omega}_b$ ). Earth rotation rate ( $\bar{\omega}_e$ ), and relative angular rates of the body with respect to north-east-down (NED) frame, expressed in body frame ( $\bar{\omega}_{rel}$ ) are defined as

$$\bar{V}_b = \begin{bmatrix} u \\ v \\ w \end{bmatrix}, \bar{\omega}_{rel} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}, \bar{\omega}_e = \begin{bmatrix} 0 \\ 0 \\ \omega_e \end{bmatrix}, \bar{\omega}_b = \bar{\omega}_{rel} + DCM_{bf} \bar{\omega}_e + DCM_{be} \bar{\omega}_{ned}$$

$$\bar{\omega}_{ned} = \begin{bmatrix} \dot{l} \cos \mu \\ -\dot{\mu} \\ -\dot{l} \sin \mu \end{bmatrix} = \begin{bmatrix} V_E / (N + h) \\ -V_N / (M + h) \\ V_E \bullet \tan \mu / (N + h) \end{bmatrix}$$

The rotational dynamics of the body defined in body-fixed frame are given below, where the applied moments are  $[L \ M \ N]^T$ , and the inertia tensor  $I$  is with respect to the origin O.

$$\bar{M}_b = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = I \dot{\bar{\omega}}_b + \bar{\omega}_b \times (I \bar{\omega}_b)$$

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix}$$

The integration of the rate of change of the quaternion vector is given below.

$$\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = -\frac{1}{2} \begin{bmatrix} 0 & \omega_b(1) & \omega_b(2) & \omega_b(3) \\ -\omega_b(1) & 0 & -\omega_b(3) & \omega_b(2) \\ -\omega_b(2) & \omega_b(3) & 0 & -\omega_b(1) \\ -\omega_b(3) & -\omega_b(2) & \omega_b(1) & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

# 6DoF ECEF (Quaternion)

## Dialog Box

Function Block Parameters: 6DoF ECEF (Quaternion)

6DoF EoM (ECEF) (mask) (link)

Integrate the six-degrees-of-freedom equations of motion using a quaternion representation for the orientation of the body in space.

Main Planet

Units: Metric (MKS)

Mass type: Fixed

Initial position in geodetic latitude, longitude, altitude [mu,l,h]:  
ipos\_g

Initial velocity in body axes [U,v,w]:  
ivel\_b

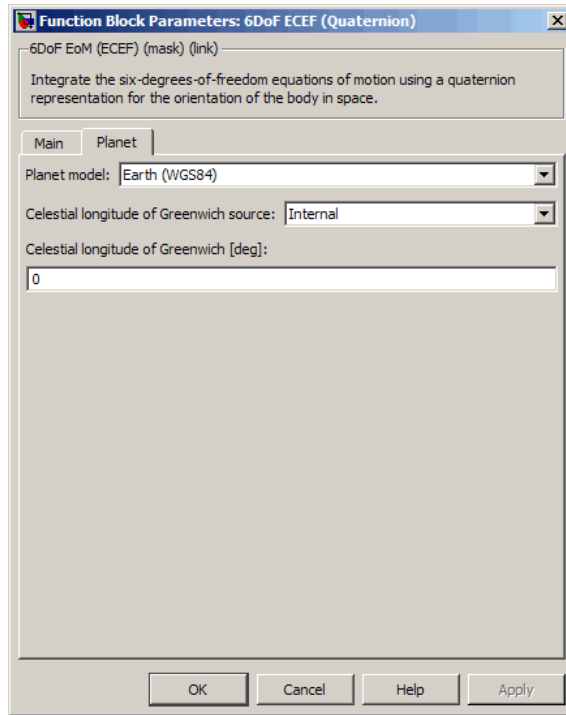
Initial Euler orientation [roll, pitch, yaw]:  
ieuler

Initial body rotation rates [p,q,r]:  
irates\_b

Initial mass:  
mass

Inertia:  
inertia

OK Cancel Help Apply



## Units

Specifies the input and output units:

# 6DoF ECEF (Quaternion)

---

<b>Units</b>	<b>Forces</b>	<b>Moment</b>	<b>Acceleration</b>	<b>Velocity</b>	<b>Position</b>	<b>Mass</b>	<b>Inertia</b>
Metric (MKS)	Newton	Newton meter	Meters per second squared	Meters per second	Meters	Kilogram	Kilogram meter squared
English (Velocity in ft/s)	Pound	Foot pound	Feet per second squared	Feet per second	Feet	Slug	Slug foot squared
English (Velocity in kts)	Pound	Foot pound	Feet per second squared	Knots	Feet	Slug	Slug foot squared

## Mass type

Select the type of mass to use:

- Fixed                      Mass is constant throughout the simulation.
- Simple Variable        Mass and inertia vary linearly as a function of mass rate (see Simple Variable Mass 6DoF ECEF (Quaternion)).
- Custom Variable        Mass and inertia variations are customizable (see Simple Variable Mass 6DoF (Quaternion)).

The Fixed selection conforms to the previously described equations of motion.

## Initial position in geodetic latitude, longitude and altitude

The three-element vector for the initial location of the body in the geodetic reference frame, in degrees.

## Initial velocity in body axes

The three-element vector containing the initial velocity of the body with respect to ECEF frame, expressed in body frame..

### **Initial Euler orientation**

The three-element vector containing the initial Euler rotation angles [roll, pitch, yaw], in radians. Euler rotation angles are those between the body and north-east-down (NED) coordinate systems.

### **Initial body rotation rates**

The three-element vector for the initial angular rates of the body with respect to NED frame, expressed in body frame, in radians per second.

### **Initial mass**

The mass of the rigid body.

### **Inertia**

The 3-by-3 inertia tensor matrix  $I$ , in body-fixed axes.

### **Planet model**

Specifies the planet model to use: Custom or Earth (WGS84).

### **Flattening**

Specifies the flattening of the planet. This option is only available when **Planet model** is set to Custom.

### **Equatorial radius of planet**

Specifies the radius of the planet at its equator. The units of the equatorial radius parameter should be the same as the units for ECEF position. This option is only available when **Planet model** is set to Custom.

### **Rotational rate**

Specifies the scalar rotational rate of the planet in rad/s. This option is only available when **Planet model** is set to Custom.

### **Celestial longitude of Greenwich source**

Specifies the source of Greenwich meridian's initial celestial longitude:

## 6DoF ECEF (Quaternion)

Internal	Use celestial longitude value from mask dialog.
External	Use external input for celestial longitude value.

### Celestial longitude of Greenwich

The initial angle between Greenwich meridian and the  $x$ -axis of the ECI frame.

### Inputs and Outputs

Input	Dimension Type	Description
First	Vector	Contains the three applied forces in body-fixed axes.
Second	Vector	Contains the three applied moments in body-fixed axes.

Output	Dimension Type	Description
First	Vector	Contains the velocity of the body with respect to ECEF frame, expressed in ECEF frame.
Second	Three-element vector	Contains the position in ECEF reference frame.
Third	Three-element vector	Contains the position in geodetic latitude, longitude and altitude, in degrees, degrees and selected units of length respectively.



Output	Dimension Type	Description
Fourth	Three-element vector	Contains the body rotation angles [roll, pitch, yaw], in radians. Euler rotation angles are those between the body and north-east-down (NED) coordinate systems.
Fifth	3-by-3 matrix	Applies to the coordinate transformation from ECI axes to body-fixed axes
Sixth	3-by-3 matrix	Applies to the coordinate transformation from NED axes to body-fixed axes.
Seventh	3-by-3 matrix	Applies to the coordinate transformation from ECEF axes to NED axes.
Eighth	Three-element vector	Contains the velocity of the body with respect to ECEF frame, expressed in the body frame.
Ninth	Three-element vector	Contains the relative angular rates of the body with respect to NED frame, expressed in the body frame, in radians per second.
Tenth	Three-element vector	Contains the angular rates of the body with respect to the ECI frame, expressed in body frame, in radians per second.
Eleventh	Three-element vector	Contains the angular accelerations of the body with respect to ECI frame, expressed in the body frame, in radians per second squared.
Twelfth	Three-element vector	Contains the accelerations in body-fixed axes.

## 6DoF ECEF (Quaternion)

---

### Assumptions and Limitations

This implementation assumes that the applied forces are acting at the center of gravity of the body, and that the mass and inertia are constant.

This implementation generates a geodetic latitude that lies between  $\pm 90$  degrees, and longitude that lies between  $\pm 180$  degrees. Additionally, the MSL altitude is approximate.

The Earth is assumed to be ellipsoidal. By setting flattening to 0.0, a spherical planet can be achieved. The Earth's precession, nutation, and polar motion are neglected. The celestial longitude of Greenwich is Greenwich Mean Sidereal Time (GMST) and provides a rough approximation to the sidereal time.

The implementation of the ECEF coordinate system assumes that the origin is at the center of the planet, the  $x$ -axis intersects the Greenwich meridian and the equator, the  $z$ -axis is the mean spin axis of the planet, positive to the north, and the  $y$ -axis completes the right-handed system.

The implementation of the ECI coordinate system assumes that the origin is at the center of the planet, the  $x$ -axis is the continuation of the line from the center of the Earth through the center of the Sun toward the vernal equinox, the  $z$ -axis points in the direction of the mean equatorial plane's north pole, positive to the north, and the  $y$ -axis completes the right-handed system.

### References

Stevens, B. L., and F. L. Lewis, *Aircraft Control and Simulation, Second Edition*, John Wiley & Sons, New York, 2003.

McFarland, Richard E., *A Standard Kinematic Model for Flight simulation at NASA-Ames*, NASA CR-2497.

"Supplement to Department of Defense World Geodetic System 1984 Technical Report: Part I - Methods, Techniques and Data Used in WGS84 Development," DMA TR8350.2-A.

### See Also

6DoF (Euler Angles)

6DoF (Quaternion)

6DoF Wind (Quaternion)

6DoF Wind (Wind Angles)

6th Order Point Mass (Coordinated Flight)

Custom Variable Mass 6DoF (Euler Angles)

Custom Variable Mass 6DoF (Quaternion)

Custom Variable Mass 6DoF ECEF (Quaternion)

Custom Variable Mass 6DoF Wind (Quaternion)

Custom Variable Mass 6DoF Wind (Wind Angles)

Simple Variable Mass 6DoF (Euler Angles)

Simple Variable Mass 6DoF (Quaternion)

Simple Variable Mass 6DoF ECEF (Quaternion)

Simple Variable Mass 6DoF Wind (Quaternion)

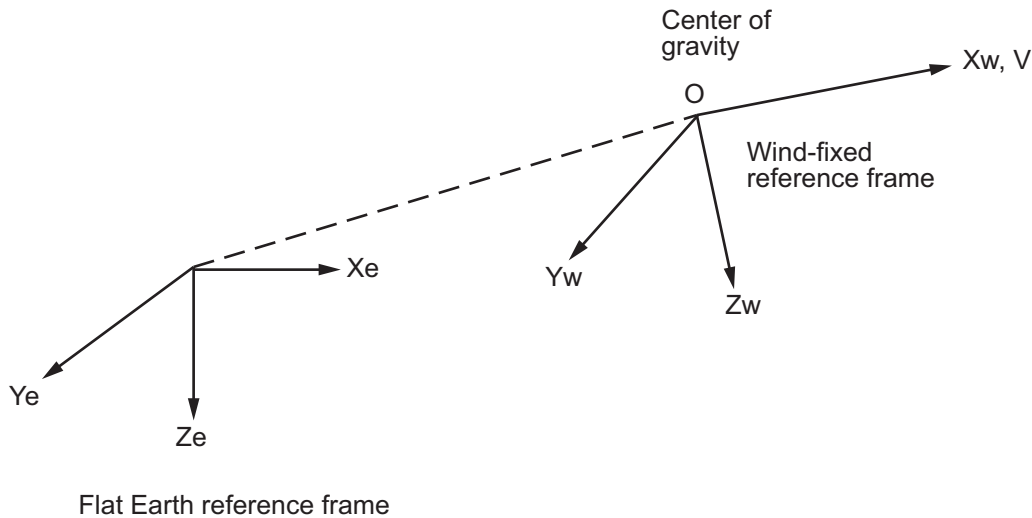
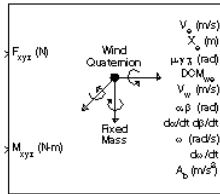
Simple Variable Mass 6DoF Wind (Wind Angles)

# 6DoF Wind (Quaternion)

**Purpose** Implement quaternion representation of six-degrees-of-freedom equations of motion with respect to wind axes

**Library** Equations of Motion/6DoF

**Description** The 6DoF Wind (Quaternion) block considers the rotation of a wind-fixed coordinate frame ( $X_w, Y_w, Z_w$ ) about an flat Earth reference frame ( $X_e, Y_e, Z_e$ ). The origin of the wind-fixed coordinate frame is the center of gravity of the body, and the body is assumed to be rigid, an assumption that eliminates the need to consider the forces acting between individual elements of mass. The flat Earth reference frame is considered inertial, an excellent approximation that allows the forces due to the Earth's motion relative to the "fixed stars" to be neglected.



The translational motion of the wind-fixed coordinate frame is given below, where the applied forces  $[F_x, F_y, F_z]^T$  are in the wind-fixed frame, and the mass of the body  $m$  is assumed constant.

$$\bar{\mathbf{F}}_w = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = m(\dot{\bar{\mathbf{V}}}_w + \bar{\boldsymbol{\omega}}_w \times \bar{\mathbf{V}}_w)$$

$$\bar{\mathbf{V}}_w = \begin{bmatrix} V \\ 0 \\ 0 \end{bmatrix}, \bar{\boldsymbol{\omega}}_w = \begin{bmatrix} p_w \\ q_w \\ r_w \end{bmatrix} = DMC_{wb} \begin{bmatrix} p_b - \dot{\beta} \sin \alpha \\ q_b - \dot{\alpha} \\ r_b + \dot{\beta} \cos \alpha \end{bmatrix}, \bar{\boldsymbol{\omega}}_b = \begin{bmatrix} p_b \\ q_b \\ r_b \end{bmatrix}$$

The rotational dynamics of the body-fixed frame are given below, where the applied moments are  $[L \ M \ N]^T$ , and the inertia tensor  $I$  is with respect to the origin O. Inertia tensor  $I$  is much easier to define in body-fixed frame.

$$\bar{\mathbf{M}}_b = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = I\dot{\bar{\boldsymbol{\omega}}}_b + \bar{\boldsymbol{\omega}}_b \times (I\bar{\boldsymbol{\omega}}_b)$$

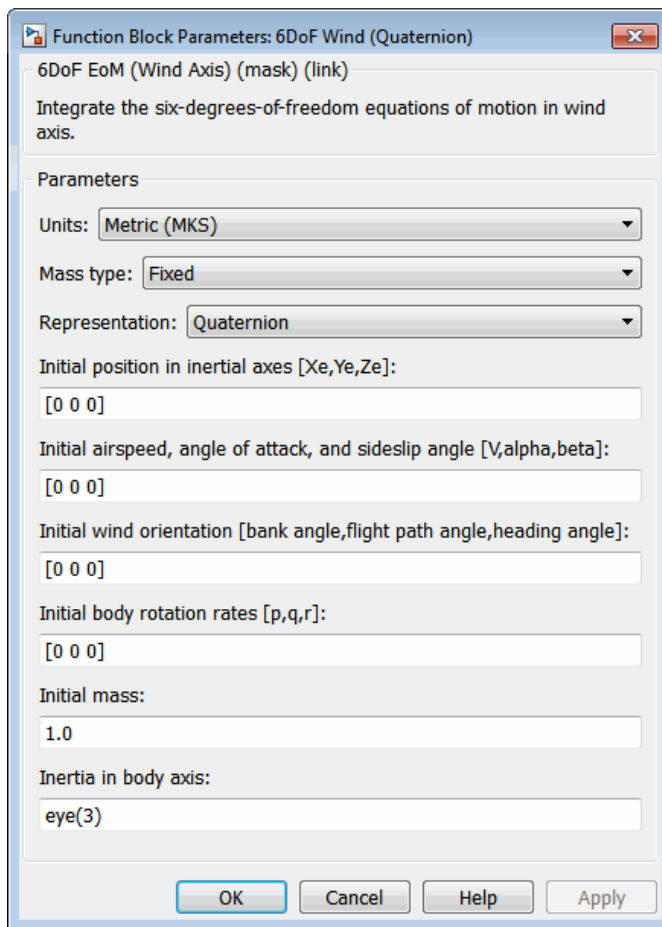
$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix}$$

The integration of the rate of change of the quaternion vector is given below.

$$\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = -\frac{1}{2} \begin{bmatrix} 0 & p & q & r \\ -p & 0 & -r & q \\ -q & r & 0 & -p \\ -r & -q & p & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

# 6DoF Wind (Quaternion)

## Dialog Box



### Units

Specifies the input and output units:

# 6DoF Wind (Quaternion)

<b>Units</b>	<b>Forces</b>	<b>Moment</b>	<b>Acceleration</b>	<b>Velocity</b>	<b>Position</b>	<b>Mass</b>	<b>Inertia</b>
Metric (MKS)	Newton	Newton meter	Meters per second squared	Meters per second	Meters	Kilogram	Kilogram meter squared
English (Velocity in ft/s)	Pound	Foot pound	Feet per second squared	Feet per second	Feet	Slug	Slug foot squared
English (Velocity in kts)	Pound	Foot pound	Feet per second squared	Knots	Feet	Slug	Slug foot squared

## Mass Type

Select the type of mass to use:

- |                 |  |
|-----------------|--|
| Fixed           | Mass is constant throughout the simulation.                |
| Simple Variable | Mass and inertia vary linearly as a function of mass rate. |
| Custom Variable | Mass and inertia variations are customizable.              |

The Fixed selection conforms to the previously described equations of motion.

## Representation

Select the representation to use:

- |             |   |
|-------------|---|
| Wind Angles | Use wind angles within equations of motion. |
| Quaternion  | Use quaternions within equations of motion. |

# 6DoF Wind (Quaternion)

---

The Quaternion selection conforms to the previously described equations of motion.

### **Initial position in inertial axes**

The three-element vector for the initial location of the body in the flat Earth reference frame.

### **Initial airspeed, angle of attack, and sideslip angle**

The three-element vector containing the initial airspeed, initial angle of attack and initial sideslip angle.

### **Initial wind orientation**

The three-element vector containing the initial wind angles [bank, flight path, and heading], in radians.

### **Initial body rotation rates**

The three-element vector for the initial body-fixed angular rates, in radians per second.

### **Initial mass**

The mass of the rigid body.

### **Inertia matrix**

The 3-by-3 inertia tensor matrix  $I$ , in body-fixed axes.

## **Inputs and Outputs**

<b>Input</b>	<b>Dimension Type</b>	<b>Description</b>
First	Vector	Contains the three applied forces in wind-fixed axes.
Second	Vector	Contains the three applied moments in body-fixed axes.

<b>Output</b>	<b>Dimension Type</b>	<b>Description</b>
First	Three-element vector	Contains the velocity in the flat Earth reference frame.
Second	Three-element vector	Contains the position in the flat Earth reference frame.



Output	Dimension Type	Description
Third	Three-element vector	Contains the wind rotation angles [bank, flight path, heading], in radians.
Fourth	3-by-3 matrix	Contains the coordinate transformation from flat Earth axes to wind-fixed axes.
Fifth	Three-element vector	Contains the velocity in the wind-fixed frame.
Sixth	Two-element vector	Contains the angle of attack and sideslip angle, in radians.
Seventh	Two-element vector	Contains the rate of change of angle of attack and rate of change of sideslip angle, in radians per second.
Eight	Three-element vector	Contains the angular rates in body-fixed axes, in radians per second.
Ninth	Three-element vector	Contains the angular accelerations in body-fixed axes, in radians per second squared.
Tenth	Three-element vector	Contains the accelerations in body-fixed axes.

### Assumptions and Limitations

The block assumes that the applied forces are acting at the center of gravity of the body, and that the mass and inertia are constant.

### Reference

Stevens, B. L., and F. L. Lewis, *Aircraft Control and Simulation*, John Wiley & Sons, New York, 1992.

### See Also

6DoF (Euler Angles)  
6DoF (Quaternion)

## 6DoF Wind (Quaternion)

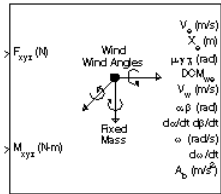
---

6DoF ECEF (Quaternion)  
6DoF Wind (Wind Angles)  
6th Order Point Mass (Coordinated Flight)  
Custom Variable Mass 6DoF (Euler Angles)  
Custom Variable Mass 6DoF (Quaternion)  
Custom Variable Mass 6DoF ECEF (Quaternion)  
Custom Variable Mass 6DoF Wind (Quaternion)  
Custom Variable Mass 6DoF Wind (Wind Angles)  
Simple Variable Mass 6DoF (Euler Angles)  
Simple Variable Mass 6DoF (Quaternion)  
Simple Variable Mass 6DoF ECEF (Quaternion)  
Simple Variable Mass 6DoF Wind (Quaternion)  
Simple Variable Mass 6DoF Wind (Wind Angles)

**Purpose** Implement wind angle representation of six-degrees-of-freedom equations of motion

**Library** Equations of Motion/6DoF

**Description** For a description of the coordinate system employed and the translational dynamics, see the block description for the 6DoF Wind (Quaternion) block.



The relationship between the wind angles,  $[\mu \ \gamma \ \chi]^T$ , can be determined by resolving the wind rates into the wind-fixed coordinate frame.

$$\begin{bmatrix} p_w \\ q_w \\ r_w \end{bmatrix} = \begin{bmatrix} \dot{\mu} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \mu & \sin \mu \\ 0 & -\sin \mu & \cos \mu \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\gamma} \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \mu & \sin \mu \\ 0 & -\sin \mu & \cos \mu \end{bmatrix} \begin{bmatrix} \cos \gamma & 0 & -\sin \gamma \\ 0 & 1 & 0 \\ \sin \gamma & 0 & \cos \gamma \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\chi} \end{bmatrix} \equiv J^{-1} \begin{bmatrix} \dot{\mu} \\ \dot{\gamma} \\ \dot{\chi} \end{bmatrix}$$

Inverting  $J$  then gives the required relationship to determine the wind rate vector.

$$\begin{bmatrix} \dot{\mu} \\ \dot{\gamma} \\ \dot{\chi} \end{bmatrix} = J \begin{bmatrix} p_w \\ q_w \\ r_w \end{bmatrix} = \begin{bmatrix} 1 & (\sin \mu \tan \gamma) & (\cos \mu \tan \gamma) \\ 0 & \cos \mu & -\sin \mu \\ 0 & \frac{\sin \mu}{\cos \gamma} & \frac{\cos \mu}{\cos \gamma} \end{bmatrix} \begin{bmatrix} p_w \\ q_w \\ r_w \end{bmatrix}$$

The body-fixed angular rates are related to the wind-fixed angular rate by the following equation.

$$\begin{bmatrix} p_w \\ q_w \\ r_w \end{bmatrix} = DMC_{wb} \begin{bmatrix} p_b - \dot{\beta} \sin \alpha \\ q_b - \dot{\alpha} \\ r_b + \dot{\beta} \cos \alpha \end{bmatrix}$$

## 6DoF Wind (Wind Angles)

---

Using this relationship in the wind rate vector equations, gives the relationship between the wind rate vector and the body-fixed angular rates.

$$\begin{bmatrix} \dot{\mu} \\ \dot{\gamma} \\ \dot{\chi} \end{bmatrix} = J \begin{bmatrix} p_w \\ q_w \\ r_w \end{bmatrix} = \begin{bmatrix} 1 & (\sin \mu \tan \gamma) & (\cos \mu \tan \gamma) \\ 0 & \cos \mu & -\sin \mu \\ 0 & \frac{\sin \mu}{\cos \gamma} & \frac{\cos \mu}{\cos \gamma} \end{bmatrix} DMC_{wb} \begin{bmatrix} p_b - \dot{\beta} \sin \alpha \\ q_b - \dot{\alpha} \\ r_b + \dot{\beta} \cos \alpha \end{bmatrix}$$

## Dialog Box

Function Block Parameters: 6DoF Wind (Wind Angles)

6DoF EoM (Wind Axis) (mask) (link)

Integrate the six-degrees-of-freedom equations of motion in wind axis.

Parameters

Units: Metric (MKS)

Mass type: Fixed

Representation: Wind Angles

Initial position in inertial axes [Xe,Ye,Ze]:  
[0 0 0]

Initial airspeed, angle of attack, and sideslip angle [V,alpha,beta]:  
[0 0 0]

Initial wind orientation [bank angle,flight path angle,heading angle]:  
[0 0 0]

Initial body rotation rates [p,q,r]:  
[0 0 0]

Initial mass:  
1.0

Inertia in body axis:  
eye(3)

OK Cancel Help Apply

### Units

Specifies the input and output units:

# 6DoF Wind (Wind Angles)

---

<b>Units</b>	<b>Forces</b>	<b>Moment</b>	<b>Acceleration</b>	<b>Velocity</b>	<b>Position</b>	<b>Mass</b>	<b>Inertia</b>
Metric (MKS)	Newton	Newton meter	Meters per second squared	Meters per second	Meters	Kilogram	Kilogram meter squared
English (Velocity in ft/s)	Pound	Foot pound	Feet per second squared	Feet per second	Feet	Slug	Slug foot squared
English (Velocity in kts)	Pound	Foot pound	Feet per second squared	Knots	Feet	Slug	Slug foot squared

## Mass type

Select the type of mass to use:

- Fixed                      Mass is constant throughout the simulation.
- Simple Variable        Mass and inertia vary linearly as a function of mass rate.
- Custom Variable        Mass and inertia variations are customizable.

The Fixed selection conforms to the previously described equations of motion.

## Representation

Select the representation to use:

- Wind Angles            Use wind angles within equations of motion.
- Quaternion            Use quaternions within equations of motion.

The Wind Angles selection conforms to the previously described equations of motion.

### **Initial position in inertial axes**

The three-element vector for the initial location of the body in the flat Earth reference frame.

### **Initial airspeed, angle of attack, and sideslip angle**

The three-element vector containing the initial airspeed, initial angle of attack and initial sideslip angle.

### **Initial wind orientation**

The three-element vector containing the initial wind angles [bank, flight path, and heading], in radians.

### **Initial body rotation rates**

The three-element vector for the initial body-fixed angular rates, in radians per second.

### **Initial mass**

The mass of the rigid body.

### **Inertia**

The 3-by-3 inertia tensor matrix  $I$ , in body-fixed axes.

## **Inputs and Outputs**

<b>Input</b>	<b>Dimension Type</b>	<b>Description</b>
First	Vector	Contains the three applied forces in wind-fixed axes.
Second	Vector	Contains the three applied moments in body-fixed axes.

<b>Output</b>	<b>Dimension Type</b>	<b>Description</b>
First	Three-element vector	Contains the velocity in the flat Earth reference frame.
Second	Three-element vector	Contains the position in the flat Earth reference frame.

## 6DoF Wind (Wind Angles)

---

Output	Dimension Type	Description
Third	Three-element vector	Contains the wind rotation angles [bank, flight path, heading], in radians.
Fourth	3-by-3 matrix	Contains the coordinate transformation from flat Earth axes to wind-fixed axes.
Fifth	Three-element vector	Contains the velocity in the wind-fixed frame.
Sixth	Two-element vector	Contains the angle of attack and sideslip angle, in radians.
Seventh	Two-element vector	Contains the rate of change of angle of attack and rate of change of sideslip angle, in radians per second.
Eighth	Three-element vector	Contains the angular rates in body-fixed axes, in radians per second.
Ninth	Three-element vector	Contains the angular accelerations in body-fixed axes, in radians per second squared.
Tenth	Three-element vector	Contains the accelerations in body-fixed axes.

### Assumptions and Limitations

The block assumes that the applied forces are acting at the center of gravity of the body, and that the mass and inertia are constant.

### Reference

Stevens, B. L., and F. L. Lewis, *Aircraft Control and Simulation*, John Wiley & Sons, New York, 1992.

### See Also

6DoF (Euler Angles)  
6DoF (Quaternion)



6DoF ECEF (Quaternion)

6DoF Wind (Quaternion)

6th Order Point Mass (Coordinated Flight)

Custom Variable Mass 6DoF (Euler Angles)

Custom Variable Mass 6DoF (Quaternion)

Custom Variable Mass 6DoF ECEF (Quaternion)

Custom Variable Mass 6DoF Wind (Quaternion)

Custom Variable Mass 6DoF Wind (Wind Angles)

Simple Variable Mass 6DoF (Euler Angles)

Simple Variable Mass 6DoF (Quaternion)

Simple Variable Mass 6DoF ECEF (Quaternion)

Simple Variable Mass 6DoF Wind (Quaternion)

Simple Variable Mass 6DoF Wind (Wind Angles)

# 6th Order Point Mass (Coordinated Flight)

## Purpose

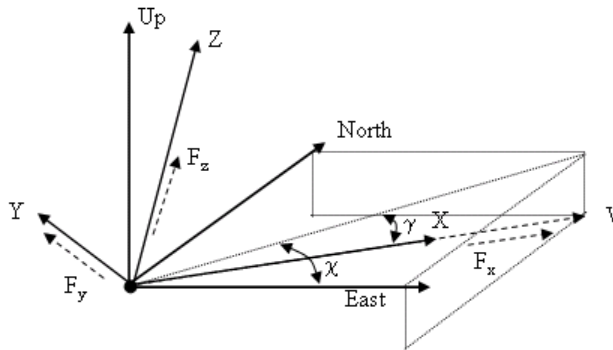
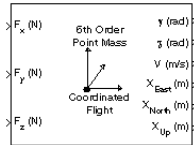
Calculate sixth-order point mass in coordinated flight

## Library

Equations of Motion/Point Mass

## Description

The 6th Order Point Mass (Coordinated Flight) block performs the calculations for the translational motion of a single point mass or multiple point masses.



The translational motion of the point mass  $[X_{East} \ X_{North} \ X_{Up}]^T$  are functions of airspeed ( $V$ ), flight path angle ( $\gamma$ ), and heading angle ( $\chi$ ),

$$\begin{aligned} F_x &= mV \\ F_y &= (mV \cos \gamma) \dot{\chi} \\ F_z &= mV \dot{\gamma} \\ \dot{X}_{East} &= V \cos \chi \cos \gamma \\ \dot{X}_{North} &= V \sin \chi \cos \gamma \\ \dot{X}_{Up} &= V \sin \gamma \end{aligned}$$

where the applied forces  $[F_x \ F_y \ F_h]^T$  are in a system is defined by  $x$ -axis in the direction of vehicle velocity relative to air,  $z$ -axis is upward, and

# 6th Order Point Mass (Coordinated Flight)

$y$ -axis completes the right-handed frame, and the mass of the body  $m$  is assumed constant.

## Dialog Box

Function Block Parameters: 6th Order Point Mass (Coordinated Flight) (mask) (link)

Calculate sixth-order point mass in coordinated flight.

Parameters

Units: Metric (MKS)

Initial flight path angle:  
0

Initial heading angle:  
0

Initial airspeed:  
100

Initial downrange [East]:  
0

Initial crossrange [North]:  
0

Initial altitude [Up]:  
0

Initial mass:  
1.0

OK Cancel Help Apply

### Units

Specifies the input and output units:

# 6th Order Point Mass (Coordinated Flight)

---

<b>Units</b>	<b>Forces</b>	<b>Velocity</b>	<b>Position</b>
Metric (MKS)	Newton	Meters per second	Meters
English (Velocity in ft/s)	Pound	Feet per second	Feet
English (Velocity in kts)	Pound	Knots	Feet

## **Initial flight path angle**

The scalar or vector containing initial flight path angle of the point mass(es).

## **Initial heading angle**

The scalar or vector containing initial heading angle of the point mass(es).

## **Initial airspeed**

The scalar or vector containing initial airspeed of the point mass(es).

## **Initial downrange [East]**

The scalar or vector containing initial downrange of the point mass(es).

## **Initial crossrange [North]**

The scalar or vector containing initial crossrange of the point mass(es).

## **Initial altitude [Up]**

The scalar or vector containing initial altitude of the point mass(es).

## **Initial mass**

The scalar or vector containing mass of the point mass(es).

# 6th Order Point Mass (Coordinated Flight)

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the force in $x$ -axis in selected units.
Second		Contains the force in $y$ -axis in selected units.
Third		Contains the force in $z$ -axis in selected units.

Output	Dimension Type	Description
First		Contains the flight path angle in radians.
Second		Contains the heading angle in radians.
Third		Contains the airspeed in selected units.
Fourth		Contains the downrange or amount traveled East in selected units.
Fifth		Contains the crossrange or amount traveled North in selected units.
Sixth		Contains the altitude or amount traveled Up in selected units.

## Assumptions and Limitations

The block assumes that there is fully coordinated flight, i.e., there is no side force (wind axes) and sideslip is always zero.

The flat flat Earth reference frame is considered inertial, an excellent approximation that allows the forces due to the Earth's motion relative to the "fixed stars" to be neglected.

## See Also

4th Order Point Mass (Longitudinal)

4th Order Point Mass Forces (Longitudinal)

6DoF (Euler Angles)

6DoF (Quaternion)

# 6th Order Point Mass (Coordinated Flight)

---

6DoF ECEF (Quaternion)  
6DoF Wind (Wind Angles)  
6th Order Point Mass Forces (Coordinated Flight)  
Custom Variable Mass 6DoF (Euler Angles)  
Custom Variable Mass 6DoF (Quaternion)  
Custom Variable Mass 6DoF ECEF (Quaternion)  
Custom Variable Mass 6DoF Wind (Quaternion)  
Custom Variable Mass 6DoF Wind (Wind Angles)  
Simple Variable Mass 6DoF (Euler Angles)  
Simple Variable Mass 6DoF (Quaternion)  
Simple Variable Mass 6DoF ECEF (Quaternion)  
Simple Variable Mass 6DoF Wind (Quaternion)  
Simple Variable Mass 6DoF Wind (Wind Angles)

# 6th Order Point Mass Forces (Coordinated Flight)

## Purpose

Calculate forces used by sixth-order point mass in coordinated flight

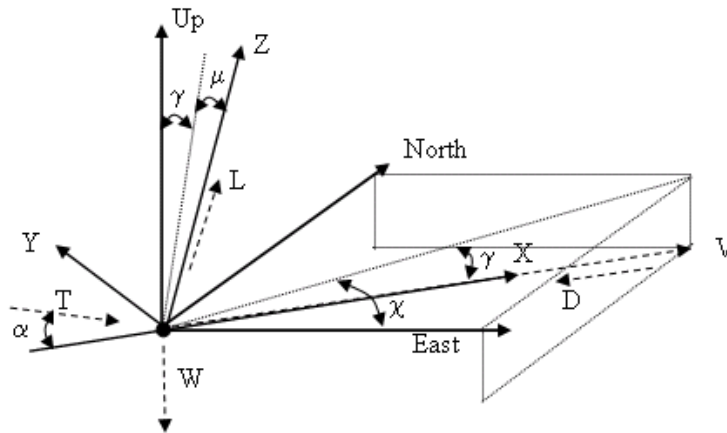
## Library

Equations of Motion/Point Mass

## Description

The 6th Order Point Mass Forces (Coordinated Flight) block calculates the applied forces for a single point mass or multiple point masses.

> Lift	$F_x$
> Drag	$F_y$
> Weight	$F_z$
> Thrust	$F_x$
> $\gamma$	$F_y$
> $\mu$	$F_z$
> $\alpha$	



The applied forces  $[F_x \ F_y \ F_z]^T$  are in a system is defined by  $x$ -axis in the direction of vehicle velocity relative to air,  $z$ -axis is upwards and  $y$ -axis completes the right-handed frame and are functions of lift ( $L$ ), drag ( $D$ ), thrust ( $T$ ), weight ( $W$ ), flight path angle ( $\gamma$ ), angle of attack ( $\alpha$ ), and bank angle ( $\mu$ ).

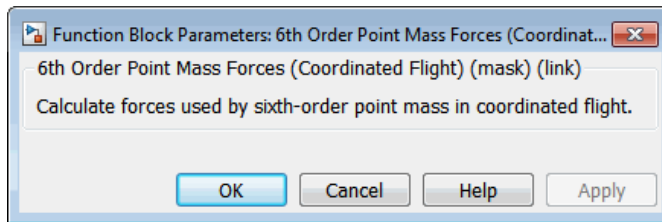
$$F_x = T \cos \alpha - D - W \sin \gamma$$

$$F_y = (L + T \sin \alpha) \sin \mu$$

$$F_z = (L + T \sin \alpha) \cos \mu - W \cos \gamma$$

# 6th Order Point Mass Forces (Coordinated Flight)

## Dialog Box



## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the lift in units of force.
Second		Contains the drag in units of force.
Third		Contains the weight in units of force.
Fourth		Contains the thrust in units of force.
Fifth		Contains the flight path angle in radians.
Sixth		Contains the bank angle in radians.
Seventh		Contains the angle of attack in radians.

Output	Dimension Type	Description
First		Contains the force in $x$ -axis in units of force.
Second		Contains the force in $y$ -axis in units of force.
Third		Contains the force in $z$ -axis in units of force.

## Assumptions and Limitations

The block assumes that there is fully coordinated flight, i.e., there is no side force (wind axes) and sideslip is always zero.

The flat Earth reference frame is considered inertial, an excellent approximation that allows the forces due to the Earth's motion relative to the "fixed stars" to be neglected.



# 6th Order Point Mass Forces (Coordinated Flight)

---

## **See Also**

4th Order Point Mass (Longitudinal)

4th Order Point Mass Forces (Longitudinal)

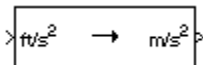
6th Order Point Mass (Coordinated Flight)

# Acceleration Conversion

**Purpose** Convert from acceleration units to desired acceleration units

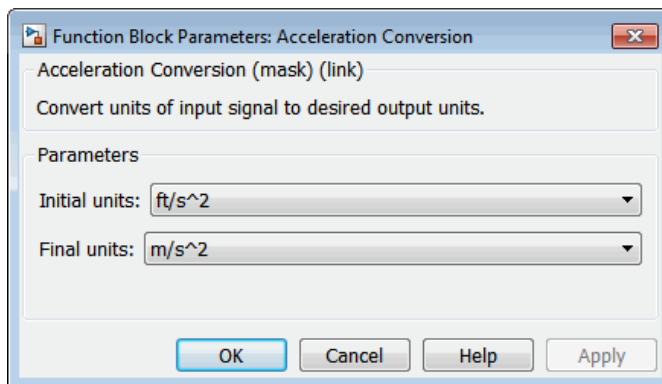
**Library** Utilities/Unit Conversions

**Description** The Acceleration Conversion block computes the conversion factor from specified input acceleration units to specified output acceleration units and applies the conversion factor to the input signal.



The Acceleration Conversion block icon displays the input and output units selected from the **Initial units** and **Final units** lists.

## Dialog Box



**Initial units**  
Specifies the input units.

**Final units**  
Specifies the output units.

The following conversion units are available:

$m/s^2$	Meters per second squared
$ft/s^2$	Feet per second squared
$km/s^2$	Kilometers per second squared

in/s <sup>2</sup>	Inches per second squared
km/h - s	Kilometers per hour per second
mph - s	Miles per hour per second
G ' s	g-units

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the acceleration in initial acceleration units.

Output	Dimension Type	Description
First		Contains the acceleration in final acceleration units.

## See Also

Angle Conversion  
Angular Acceleration Conversion  
Angular Velocity Conversion  
Density Conversion  
Force Conversion  
Length Conversion  
Mass Conversion  
Pressure Conversion  
Temperature Conversion  
Velocity Conversion

# Adjoint of 3x3 Matrix

---

**Purpose** Compute adjoint of matrix

**Library** Utilities/Math Operations

**Description** The Adjoint of 3x3 Matrix block computes the adjoint matrix for the input matrix.



The input matrix has the form of

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}$$

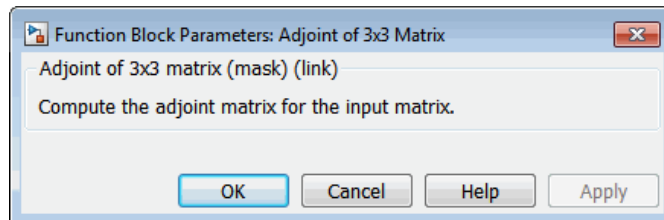
The adjoint of the matrix has the form of

$$\text{adj}(A) = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix}$$

where

$$M_{ij} = (-1)^{i+j}$$

## Dialog Box



## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the acceleration in initial acceleration units.

Output	Dimension Type	Description
First		Contains the acceleration in final acceleration units.

## See Also

[Create 3x3 Matrix](#)

[Determinant of 3x3 Matrix](#)

[Invert 3x3 Matrix](#)

# Aerodynamic Forces and Moments

## Purpose

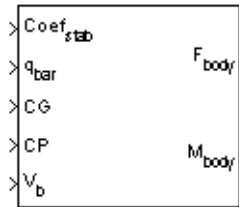
Compute aerodynamic forces and moments using aerodynamic coefficients, dynamic pressure, center of gravity, center of pressure, and velocity

## Library

Aerodynamics

## Description

The Aerodynamic Forces and Moments block computes the aerodynamic forces and moments about the center of gravity. By default, the inputs and outputs are represented in the body axes.



Let  $\alpha$  be the angle of attack and  $\beta$  the sideslip. The rotation from body to stability axes:

$$C_{s \leftarrow b} = \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix}$$

can be combined with the rotation from stability to wind axes:

$$C_{w \leftarrow s} = \begin{bmatrix} \cos(\beta) & \sin(\beta) & 0 \\ -\sin(\beta) & \cos(\beta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

to yield the net rotation from body to wind axes:

$$C_{w \leftarrow b} = \begin{bmatrix} \cos(\alpha)\cos(\beta) & \sin(\beta) & \sin(\alpha)\cos(\beta) \\ -\cos(\alpha)\sin(\beta) & \cos(\beta) & -\sin(\alpha)\sin(\beta) \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix}$$

Moment coefficients have the same notation in all systems. Force coefficients are given below. Note there are no specific symbols for stability-axes force components. However, the stability axes have two components that are unchanged from the other axes.

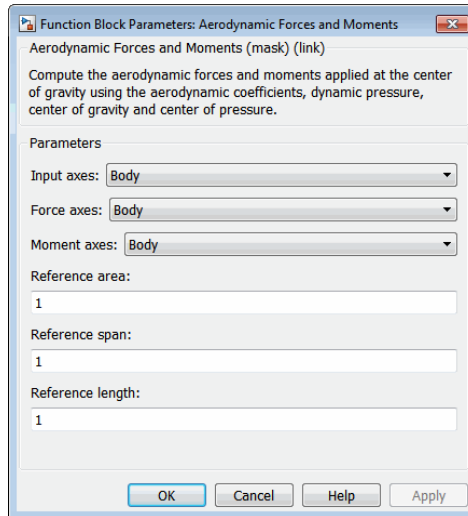
# Aerodynamic Forces and Moments

$$\mathbf{F}_A^w \equiv \begin{bmatrix} -D \\ -C \\ -L \end{bmatrix} = C_{w \leftarrow b} \cdot \begin{bmatrix} X_A \\ Y_A \\ Z_A \end{bmatrix} \equiv C_{w \leftarrow b} \cdot \mathbf{F}_A^b$$

Components/Axes	<b>x</b>	<b>y</b>	<b>z</b>
Wind	$C_D$	$C_C$	$C_L$
Stability	—	$C_Y$	$C_L$
Body	$C_X$	$C_Y$	$C_Z (-C_N)$

Given these definitions, to account for the standard definitions of  $D$ ,  $C$ ,  $Y$  (where  $Y = -C$ ), and  $L$ , force coefficients in the wind axes are multiplied by the negative identity  $diag(-1, -1, -1)$ . Forces coefficients in the stability axes are multiplied by  $diag(-1, 1, -1)$ .  $C_N$  and  $C_X$  are, respectively, the normal and axial force coefficients ( $C_N = -C_Z$ ).

## Dialog Box



# Aerodynamic Forces and Moments

---

## Input Axes

Specifies coordinate system for input coefficients: **Body** (default), **Stability**, or **Wind**.

## Force Axes

Specifies coordinate system for aerodynamic force: **Body** (default), **Stability**, or **Wind**.

## Moment Axes

Specifies coordinate system for aerodynamic moment: **Body** (default), **Stability**, or **Wind**.

## Reference area

Specifies the reference area for calculating aerodynamic forces and moments.

## Reference span

Specifies the reference span for calculating aerodynamic moments in  $x$ -axes and  $z$ -axes.

## Reference length

Specifies the reference length for calculating aerodynamic moment in the  $y$ -axes.

## Inputs and Outputs

The first input consists of aerodynamic coefficients (in the chosen input axes) for forces and moments. These coefficients are ordered into a vector depending on the choice of axes:

Input Axes	Input Vector
Body	(axial force $C_x$ , side force $C_y$ , normal force $C_z$ , rolling moment $C_l$ , pitching moment $C_m$ , yawing moment $C_n$ )
Stability	(drag force $C_{D(\beta=0)}$ , side force $C_y$ , lift force $C_L$ , rolling moment $C_l$ , pitching moment $C_m$ , yawing moment $C_n$ )
Wind	(drag force $C_D$ , cross-wind force $C_c$ , lift force $C_L$ , rolling moment $C_l$ , pitching moment $C_m$ , yawing moment $C_n$ )



# Aerodynamic Forces and Moments

Input	Dimension Type	Description
Second		Contains the dynamic pressure.
Third		Contains the center of gravity.
Fourth		Contains the center of pressure. This can also be taken as any general moment reference point as long as the rest of the model reflects the use of the moment reference point.
Fifth (For inputs or outputs in stability or wind axes)	Three-element vector	Contains the velocity in the body axes.

Output	Dimension Type	Description
First		Contains the aerodynamic forces (in the chosen output axes) at the center of gravity in $x$ -, $y$ -, and $z$ -axes.
Second		Contains the aerodynamic moments (in the chosen output axes) at the center of gravity in $x$ -, $y$ -, and $z$ -axes.

## Assumptions and Limitations

The default state of the block hides the  $V_b$  input port and assumes that the transformation is body-body.

The center of gravity and the center of pressure are assumed to be in body axes.

# Aerodynamic Forces and Moments

---

While this block has the ability to output forces and/or moments in the stability axes, the blocks in the Equations of Motion library are currently designed to accept forces and moments in either the body or wind axes only.

## **Examples**

See Airframe in aeroblk\_HL20 for an example of this block.

## **Reference**

Stevens, B. L., and F. L. Lewis, *Aircraft Control and Simulation*, John Wiley & Sons, New York, 1992

## **See Also**

Digital DATCOM Forces and Moments

Dynamic Pressure

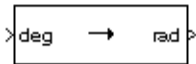
Estimate Center of Gravity

Moments About CG Due to Forces

**Purpose** Convert from angle units to desired angle units

**Library** Utilities/Unit Conversions

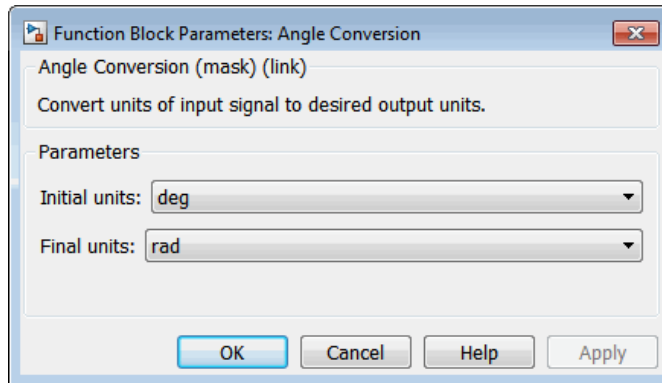
## Description



The Angle Conversion block computes the conversion factor from specified input angle units to specified output angle units and applies the conversion factor to the input signal.

The Angle Conversion block icon displays the input and output units selected from the **Initial units** and the **Final units** lists.

## Dialog Box



**Initial units**  
Specifies the input units.

**Final units**  
Specifies the output units.

The following conversion units are available:

deg	Degrees
rad	Radians
rev	Revolutions

# Angle Conversion

---

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the angle in initial angle units.

Output	Dimension Type	Description
First		Contains the angle in final angle units.

## See Also

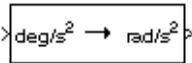
Acceleration Conversion  
Angular Acceleration Conversion  
Angular Velocity Conversion  
Density Conversion  
Force Conversion  
Length Conversion  
Mass Conversion  
Pressure Conversion  
Temperature Conversion  
Velocity Conversion

# Angular Acceleration Conversion

**Purpose** Convert from angular acceleration units to desired angular acceleration units

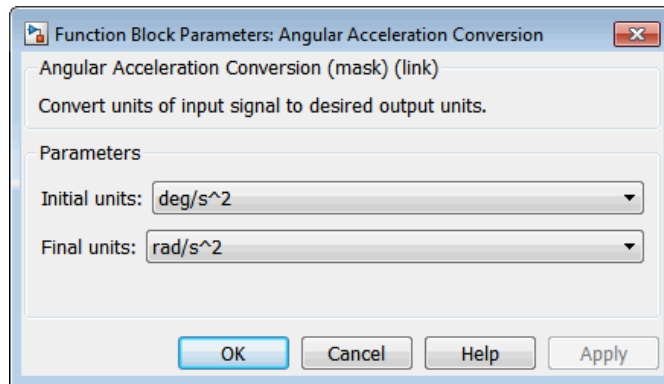
**Library** Utilities/Unit Conversions

**Description** The Angular Acceleration Conversion block computes the conversion factor from specified input angular acceleration units to specified output angular acceleration units and applies the conversion factor to the input signal.



The Angular Acceleration Conversion block icon displays the input and output units selected from the **Initial units** and the **Final units** lists.

## Dialog Box



**Initial units**  
Specifies the input units.

**Final units**  
Specifies the output units.

The following conversion units are available:

# Angular Acceleration Conversion

---

deg/s <sup>2</sup>	Degrees per second squared
rad/s <sup>2</sup>	Radians per second squared
rpm/s	Revolutions per minute per second

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the angular acceleration in initial angular acceleration units.

Output	Dimension Type	Description
First		Contains the angular acceleration in final angular acceleration units.

## See Also

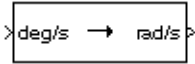
Acceleration Conversion  
Angle Conversion  
Angular Velocity Conversion  
Density Conversion  
Force Conversion  
Length Conversion  
Mass Conversion  
Pressure Conversion  
Temperature Conversion  
Velocity Conversion

# Angular Velocity Conversion

**Purpose** Convert from angular velocity units to desired angular velocity units

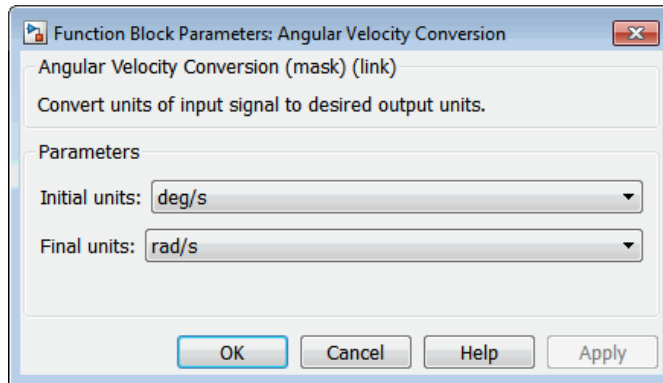
**Library** Utilities/Unit Conversions

**Description** The Angular Velocity Conversion block computes the conversion factor from specified input angular velocity units to specified output angular velocity units and applies the conversion factor to the input signal.



The Angular Velocity Conversion block icon displays the input and output units selected from the **Initial units** and the **Final units** lists.

## Dialog Box



**Initial units**  
Specifies the input units.

**Final units**  
Specifies the output units.

The following conversion units are available:

deg/s	Degrees per second
rad/s	Radians per second
rpm	Revolutions per minute

# Angular Velocity Conversion

---

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the angular acceleration in initial angular acceleration units.

Output	Dimension Type	Description
First		Contains the angular acceleration in final angular acceleration units.

## See Also

Acceleration Conversion  
Angle Conversion  
Angular Acceleration Conversion  
Density Conversion  
Force Conversion  
Length Conversion  
Mass Conversion  
Pressure Conversion  
Temperature Conversion  
Velocity Conversion



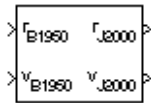
## Purpose

Transform position and velocity components from discontinued Standard Besselian Epoch (B1950) to Standard Julian Epoch (J2000)

## Library

Utilities/Axes Transformations

## Description



The Besselian Epoch to Julian Epoch block transforms two 3-by-1 vectors of Besselian Epoch position ( $\bar{r}_{B1950}$ ), and Besselian Epoch velocity ( $\bar{v}_{B1950}$ ) into Julian Epoch position ( $\bar{r}_{J2000}$ ), and Julian Epoch velocity ( $\bar{v}_{J2000}$ ). The transformation is calculated using:

$$\begin{bmatrix} \bar{r}_{J2000} \\ \bar{v}_{J2000} \end{bmatrix} = \begin{bmatrix} \bar{M}_{rr} & \bar{M}_{vr} \\ \bar{M}_{rv} & \bar{M}_{vv} \end{bmatrix} \begin{bmatrix} \bar{r}_{B1950} \\ \bar{v}_{B1950} \end{bmatrix}$$

where  $(\bar{M}_{rr}, \bar{M}_{vr}, \bar{M}_{rv}, \bar{M}_{vv})$  are defined as:

$$\bar{M}_{rr} = \begin{bmatrix} 0.9999256782 & -0.0111820611 & -0.0048579477 \\ 0.0111820610 & 0.9999374784 & -0.0000271765 \\ 0.0048579479 & -0.0000271474 & 0.9999881997 \end{bmatrix}$$

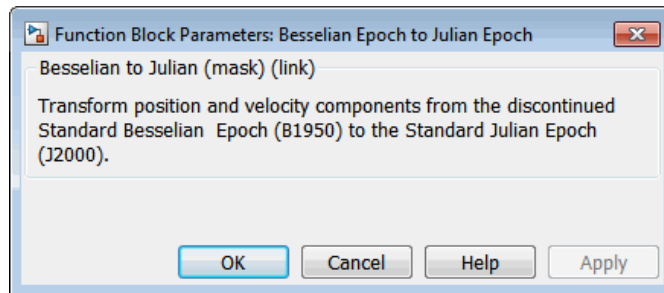
$$\bar{M}_{vr} = \begin{bmatrix} 0.00000242395018 & -0.00000002710663 & -0.00000001177656 \\ 0.00000002710663 & 0.00000242397878 & -0.0000000006587 \\ 0.00000001177656 & -0.0000000006582 & 0.00000242410173 \end{bmatrix}$$

$$\bar{M}_{rv} = \begin{bmatrix} -0.000551 & -0.238565 & 0.435739 \\ 0.238514 & -0.002667 & -0.008541 \\ -0.435623 & 0.012254 & 0.002117 \end{bmatrix}$$

$$\bar{M}_{vv} = \begin{bmatrix} 0.99994704 & -0.01118251 & -0.00485767 \\ 0.01118251 & 0.99995883 & -0.00002718 \\ 0.00485767 & -0.00002714 & 1.00000956 \end{bmatrix}$$

# Besselian Epoch to Julian Epoch

## Dialog Box



## Inputs and Outputs

Input	Dimension Type	Description
First	3-by-1 vector	Contains the position in Standard Besselian Epoch (B1950).
Second	3-by-1 vector	Contains the velocity in Standard Besselian Epoch (B1950).

Output	Dimension Type	Description
First	3-by-1 vector	Contains the position in Standard Julian Epoch (J2000).
Second	3-by-1 vector	Contains the velocity in Standard Julian Epoch (J2000).

## Reference

“Supplement to Department of Defense World Geodetic System 1984 Technical Report: Part I - Methods, Techniques and Data Used in WGS84 Development,” DMA TR8350.2-A.

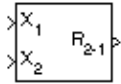
## See Also

Julian Epoch to Besselian Epoch

**Purpose** Calculate range between two crafts given their respective positions

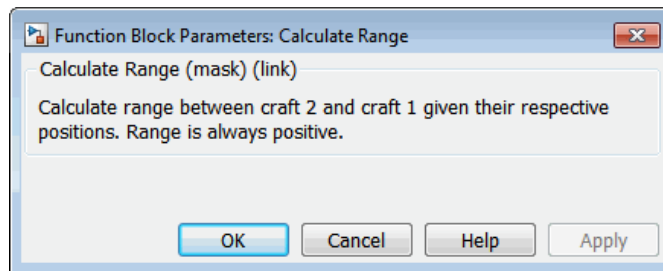
**Library** GNC/Guidance

**Description** The Calculate Range block computes the range between two crafts. The equation used for the range calculation is



$$Range = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

## Dialog Box



## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the (x, y and z) position of craft 1.
Second		Contains the (x, y and z) position of craft 2.

Output	Dimension Type	Description
First		Contains the range from craft 2 and craft 1.

**Limitation** The calculated range is the magnitude of the distance, but not the direction. Therefore it is always positive or zero.

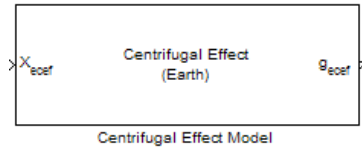
Craft positions are real values.

# Centrifugal Effect Model

**Purpose** Implement mathematical representation of centrifugal effect for planetary gravity

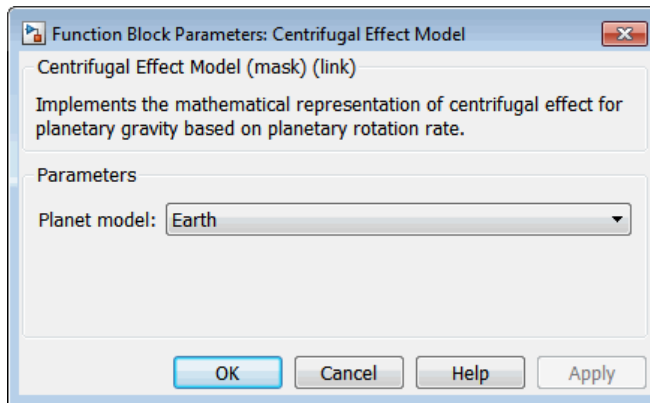
**Library** Environment/Gravity

## Description



The Centrifugal Effect Model block implements the mathematical representation of centrifugal effect for planetary gravity. The gravity centrifugal effect is the acceleration portion of centrifugal force effects due to the rotation of a planet. This block implements this representation using planetary rotation rates. You use centrifugal force values in rotating or non-inertial coordinate systems.

## Dialog Box



### Planet model

Specify the planetary model. From the list, select Mercury, Venus, Earth, Moon, Mars, Jupiter, Saturn, Uranus, Neptune, or Custom.

The block uses the rotation of the selected planet to implement the mathematical representation of the centrifugal effect.

Selecting Custom enables you to specify your own planetary model. This option enables the **Planetary rotational rate (rad/sec)** and **Input planetary rotation rate** parameters.

### **Planetary rotational rate (rad/sec)**

Specify the planetary rotational rate in radians per second.

If you want to specify the planetary rotational rate as an input to the block, see the **Input planetary rotation rate** parameter.

Selecting the **Input planetary rotation rate** check box disables the **Planetary rotational rate (rad/sec)** parameter.

### **Input planetary rotation rate**

Select this check box to enable a block input. You can then input a planetary rotation rate as a block input. When you select this check box, the block mask updates to display an input port for the rotation rate.

Selecting the **Input planetary rotation rate** check box disables the **Planetary rotational rate (rad/sec)** parameter.

## **References**

Vallado, D. A., *Fundamentals of Astrodynamics and Applications*, McGraw-Hill, New York, 1997.

NIMA TR8350.2: *Department of Defense World Geodetic System 1984, Its Definition and Relationship with Local Geodetic Systems*.

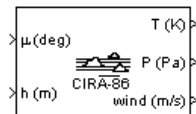
# CIRA-86 Atmosphere Model

---

**Purpose** Implement mathematical representation of 1986 CIRA atmosphere

**Library** Environment/Atmosphere

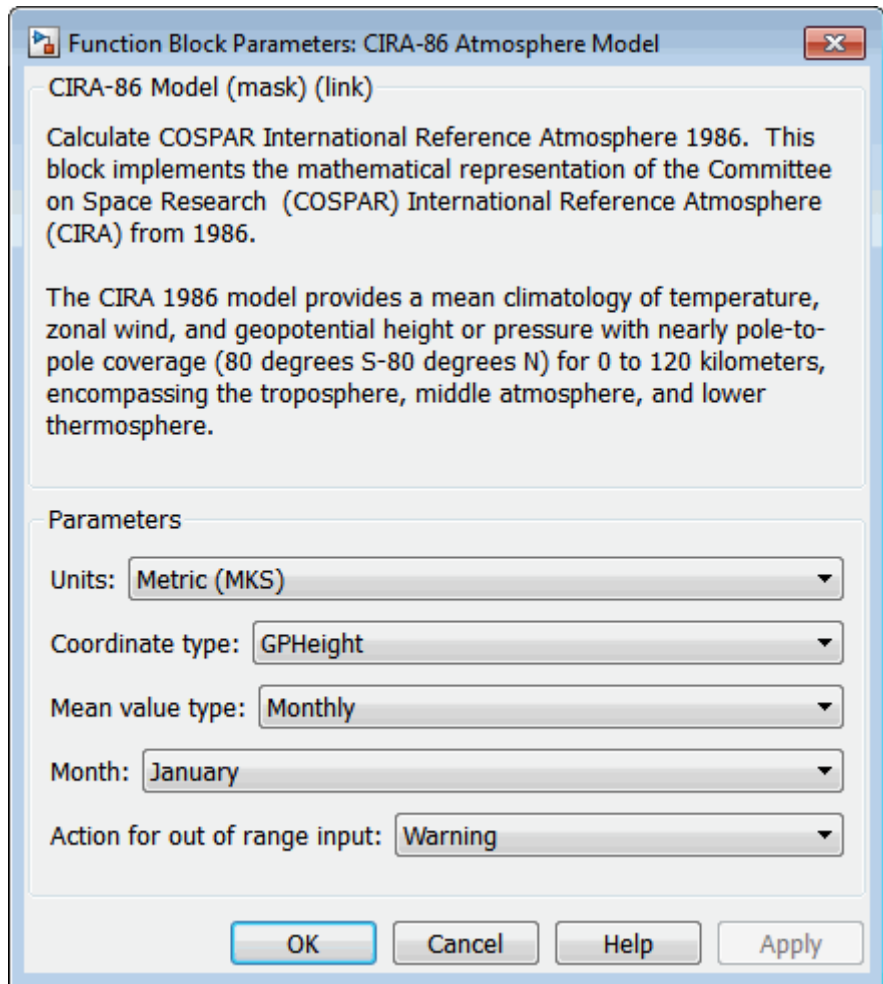
## Description



The CIRA-86 Atmosphere Model block implements the mathematical representation of the 1986 Committee on Space Research (COSPAR) International Reference Atmosphere (CIRA). The block provides values for absolute temperature, pressure, density, and speed of sound for the input geopotential altitude.

The CIRA-86 Atmosphere Model block icon displays the input and output units selected from the **Units** list.

## Dialog Box



### Units

Specifies the input and output units:

# CIRA-86 Atmosphere Model

---

<b>Units</b>	<b>Height</b>	<b>Temperature</b>	<b>Speed of Sound</b>	<b>Air Pressure</b>	<b>Air Density</b>
Metric (MKS)	Meters	Kelvin	Meters per second	Pascal	Kilograms per cubic meter
English (Velocity in ft/s)	Feet	Degrees Rankine	Feet per second	Pound-force per square inch	Slug per cubic foot
English (Velocity in kts)	Feet	Degrees Rankine	Knots	Pound-force per square inch	Slug per cubic foot

## **Coordinate type**

Specify the representation of the coordinate type. The default is GPHeight.

- Pressure  
Indicates pressure in pascal.
- GPHeight  
Indicates geopotential height in meters.

## **Mean value type**

Specify mean value types. The default is Monthly.

- Monthly  
Indicates monthly values. If you select Monthly, you must also set the **Month** parameter.
- Annual  
Indicates annual values. Valid when **Coordinate type** has a value of Pressure.



## Month

Indicates the month in which the mean values are taken. From the list, select the desired month. This parameter applies only when **Mean value type** has a value of Monthly.

## Action for out of range input

Specify if out-of-range input invokes a warning, error, or no action.

## Inputs and Outputs

Input	Dimension Type	Description
First	Array	Contains the latitude in degrees (limited to +/-80 degrees).
Second	Array	Contains an m array of either: <ul style="list-style-type: none"> <li>• Geopotential heights in selected length units (<b>Coordinate type</b> is GPHeight)</li> <li>• Pressures in selected pressure units (<b>Coordinate type</b> is Pressure)</li> </ul>

Output	Dimension Type	Description
First	Array	Contains mean temperature in selected units.
Second	Array	Contains geopotential heights in selected units.
Third	Array	Contains mean zonal winds in selected units.

## Assumptions and Limitations

This function uses a corrected version of the CIRA data files provided by J. Barnett in July 1990 in ASCII format.

This function has the limitations of the CIRA 1986 model. The values for the CIRA 1986 model are limited to the regions of 80 degrees S

# CIRA-86 Atmosphere Model

---

to 80 degrees N on the Earth and geopotential heights of 0 to 120 kilometers. In each monthly mean data set, values at 80 degrees S for 101,300 pascal or 0 meters were omitted because these levels are within the Antarctic land mass. For zonal mean pressure in constant altitude coordinates, pressure data is not available below 20 kilometers. Therefore, this is the bottom level of the CIRA climatology.

## Reference

Fleming, E. L., Chandra, S., Shoerberl, M. R., Barnett, J. J., *Monthly Mean Global Climatology of Temperature, Wind, Geopotential Height and Pressure for 0-120 km*, NASA TM100697, February 1988

<http://modelweb.gsfc.nasa.gov/atmos/cospar1.html>

## See Also

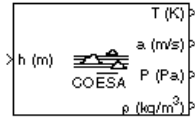
COESA Atmosphere Model

ISA Atmosphere Model

**Purpose** Implement 1976 COESA lower atmosphere

**Library** Environment/Atmosphere

## Description



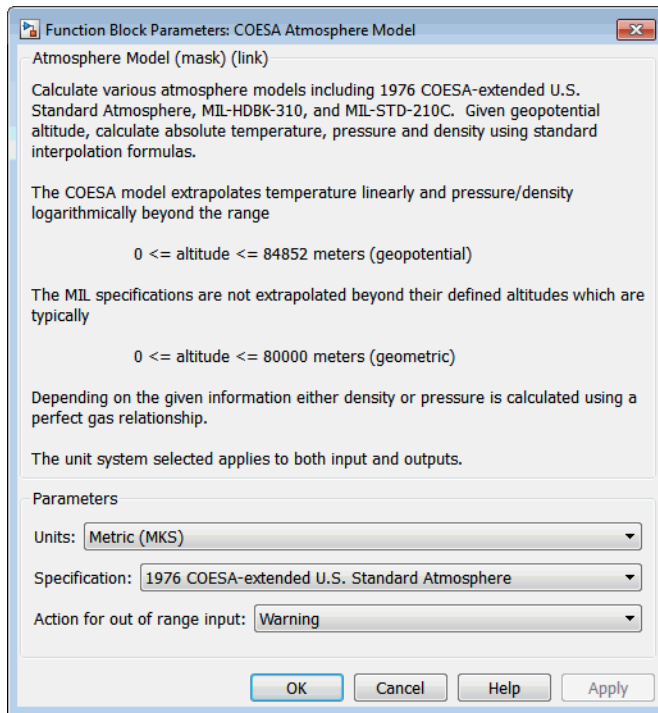
The COESA Atmosphere Model block implements the mathematical representation of the 1976 Committee on Extension to the Standard Atmosphere (COESA) United States standard lower atmospheric values for absolute temperature, pressure, density, and speed of sound for the input geopotential altitude.

Below 32,000 meters (approximately 104,987 feet), the U.S. Standard Atmosphere is identical with the Standard Atmosphere of the International Civil Aviation Organization (ICAO).

The COESA Atmosphere Model block icon displays the input and output units selected from the **Units** list.

# COESA Atmosphere Model

## Dialog Box



## Units

Specifies the input and output units:

<b>Units</b>	<b>Height</b>	<b>Temperature</b>	<b>Speed of Sound</b>	<b>Air Pressure</b>	<b>Air Density</b>
Metric (MKS)	Meters	Kelvin	Meters per second	Pascal	Kilograms per cubic meter
English (Velocity in ft/s)	Feet	Degrees Rankine	Feet per second	Pound-force per square inch	Slug per cubic foot
English (Velocity in kts)	Feet	Degrees Rankine	Knots	Pound-force per square inch	Slug per cubic foot

## Specification

Specify the atmosphere model type from one of the following atmosphere models. The default is 1976 COESA-extended U.S. Standard Atmosphere.

MIL-HDBK-310

This selection is linked to the Non-Standard Day 310 block. See the block reference for more information.

MIL-STD-210C

This selection is linked to the Non-Standard Day 210C block. See the block reference for more information.

## Action for out of range input

Specify if out-of-range input invokes a warning, error, or no action.

## Inputs and Outputs

<b>Input</b>	<b>Dimension Type</b>	<b>Description</b>
First		Contains the geopotential height.

# COESA Atmosphere Model

---

Output Dimension Type	Description
First	Contains the temperature.
Second	Contains the speed of sound.
Third	Contains the air pressure.
Fourth	Contains the air density.

## Assumptions and Limitations

Below the geopotential altitude of 0 m (0 feet) and above the geopotential altitude of 84,852 m (approximately 278,386 feet), temperature values are extrapolated linearly and pressure values are extrapolated logarithmically. Density and speed of sound are calculated using a perfect gas relationship.

## Examples

See the `aeroblk_calibrated` model, the `aeroblk_indicated` model, and the airframe in `aeroblk_HL20` for examples of this block.

## Reference

*U.S. Standard Atmosphere*, 1976, U.S. Government Printing Office, Washington, D.C.

## See Also

CIRA-86 Atmosphere Model, ISA Atmosphere Model

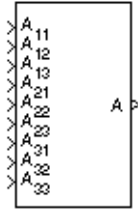
Non-Standard Day 210C

Non-Standard Day 310

**Purpose** Create 3-by-3 matrix from nine input values

**Library** Utilities/Math Operations

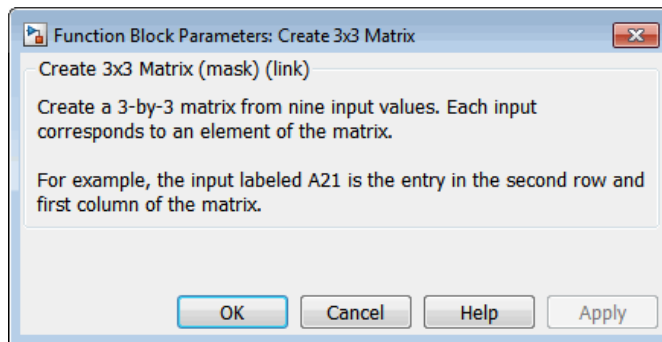
**Description** The Create 3x3 Matrix block creates a 3-by-3 matrix from nine input values where each input corresponds to an element of the matrix.



The output matrix has the form of

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}$$

## Dialog Box



## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the first row and first column of the matrix.
Second		Contains the first row and second column of the matrix.
Third		Contains the first row and third column of the matrix.

# Create 3x3 Matrix

---

Input	Dimension Type	Description
Fourth		Contains the second row and first column of the matrix.
Fifth		Contains the second row and second column of the matrix.
Sixth		Contains the second row and third column of the matrix.
Seventh		Contains the third row and first column of the matrix.
Eight		Contains the third row and second column of the matrix.
Ninth		Contains the third row and third column of the matrix.

Output	Dimension Type	Description
First	3-by-3 matrix	Contains the matrix.

## See Also

Adjoint of 3x3 Matrix

Determinant of 3x3 Matrix

Invert 3x3 Matrix

Symmetric Inertia Tensor



**Purpose** Represent crossover pilot model

**Library** Pilot Models

## Description



The Crossover Pilot Model block represents the pilot model described in *Mathematical Models of Human Pilot Behavior*. (For more information, see [1]). This pilot model is a single input, single output (SISO) model that represents some aspects of human behavior when controlling aircraft. When modeling human pilot models, use this block for more accuracy than that provided by the Tustin Pilot Model block. This block is also less accurate than the Precision Pilot Model block.

The Crossover Model takes into account the combined dynamics of the human pilot and the aircraft, using the following form around the crossover frequency:

$$Y_p Y_c = \frac{\omega_c e^{-\tau s}}{s}.$$

In this equation:

Variable	Description
$Y_p$	Pilot transfer function.
$Y_c$	Aircraft transfer function.
$\omega_c$	Crossover frequency.
$\tau$	Transport delay time caused by the pilot neuromuscular system.

If the dynamics of the aircraft ( $Y_c$ ) change,  $Y_p$  changes correspondingly. From the options provided in the **Type of control** parameter, specify the dynamics of the aircraft. The preceding table lists the possible types of control that you can select for the aircraft.

# Crossover Pilot Model

---

---

**Note** This block is valid only around the crossover frequency. It is not valid for discrete inputs such as a step.

---

This block has non-linear behavior. If you want to linearize the block (for example, with one of the Simulink `linmod` functions), you might need to change the Pade approximation order. The Crossover Pilot Model block implementation incorporates the Simulink Transport Delay block with the **Pade order (for linearization)** parameter set to 2 by default. To change this value, use the `set_param` function, for example:

```
set_param(gcf, 'pade', '3')
```

## Dialog Box

Function Block Parameters: Crossover Pilot Model

CrossoverPilotModel (mask) (link)

Implement a human pilot model for a single-input, single-output system, based on the crossover model proposed by D. T. McRuer.

(\*) Indicates that the pilot model includes a derivative block, for which additional care must be taken so the input is "smooth".

Parameters

Type of control: Proportional

Calculated value: Pilot gain

Controlled element gain:  
1

Pilot gain:  
3

Crossover frequency (rad/s):  
3

Pilot time delay (s):  
0.1

OK Cancel Help Apply

### Type of control

From the list, select one of the following options to specify the type of dynamics control that you want the pilot to have over for the aircraft.

# Crossover Pilot Model

Option (Controlled Element Transfer Function)	Transfer Function of Controlled Element ( $Y_c$ )	Transfer Function of Pilot ( $Y_p$ )	$Y_c Y_p$	Notes
Proportional	$K_c$	$\frac{K_p e^{-\tau s}}{s}$	$\frac{K_c K_p e^{-\tau s}}{s}$	
Rate or velocity	$\frac{K_c}{s}$	$K_p e^{-\tau s}$	$\frac{K_c K_p e^{-\tau s}}{s}$	
Spiral divergence	$\frac{K_c}{T_I s - 1}$	$K_p e^{-\tau s}$	$\frac{K_c K_p e^{-\tau s}}{(T_I s - 1)}$	
Second order - Short period	$\frac{K_c \omega_n^2}{s^2 + 2\zeta \omega_n s + \omega_n^2}$	$\frac{K_p e^{-\tau s}}{T_I s + 1}$	$\frac{K_c \omega_n^2}{s^2 + 2\zeta \omega_n s + \omega_n^2} \times \frac{K_p e^{-\tau s}}{T_I s + 1}$	Short period, with $\omega_n > 1/\tau$
Acceleration (*)	$\frac{K_c}{s^2}$	$K_p s e^{-\tau s}$	$\frac{K_c K_p e^{-\tau s}}{s}$	
Roll attitude (*)	$\frac{K_c}{s(T_I s + 1)}$	$K_p (T_L s + 1) e^{-\tau s}$	$\frac{K_c K_p e^{-\tau s}}{s}$	With $T_L \approx T_I$

# Crossover Pilot Model

Option (Controlled Element Transfer Function)	Transfer Function of Controlled Element ( $Y_c$ )	Transfer Function of Pilot ( $Y_p$ )	$Y_c Y_p$	Notes
Unstable short period(*)	$\frac{K_c}{(T_{I1}s + 1)(T_{I2}s - 1)}$	$K_p(T_L s + 1)e^{-\tau s}$	$\frac{K_c K_p e^{-\tau s}}{(T_{I2}s - 1)}$	With $T_L \approx T_{I1}$
Second order - Phugoid(*)	$\frac{K_c \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$	$K_p(T_L s + 1)e^{-\tau s}$	$\frac{K_c K_p \omega_n^2 e^{-\tau s}}{s}$	Phugoid, with $\omega_n \approx 1/\tau$ , $1/T_L \approx \zeta\omega_n$

\* Indicates that the pilot model includes a Derivative block, which produces a numerical derivative. For this reason, do not send discontinuous (such as a step) or noisy input to the Crossover Pilot Model block. Such inputs can cause large outputs that might render the system unstable.

Variable	Description
$K_c$	Aircraft gain.
$K_p$	Pilot gain.
$\tau$	Pilot time delay.
$T_I$	Lag constant.

# Crossover Pilot Model

---

Variable	Description
$T_L$	Lead constant.
$\zeta$	Damping ratio for the aircraft.
$\omega_n$	Natural frequency of the aircraft.

## Calculated value

From the list, select one of the following options to specify which value the block is to calculate:

- **Crossover frequency** — The block calculates the crossover frequency value. Selecting this option disables the **Crossover frequency (rad/s)** parameter.
- **Pilot gain** — The block calculates the pilot gain value. Selecting this option disables the **Pilot gain** parameter.

## Controlled element gain

Specifies the gain of the aircraft controlled dynamics.

## Pilot gain

Specifies the pilot gain.

## Crossover frequency (rad/s)

Specifies a crossover frequency value, rad/s. This value ranges from 1 to 10 rad/s.

## Pilot time delay(s)

Specifies the total pilot time delay, in seconds. This value typically ranges from 0.1 s to 0.2 s.

## Inputs and Outputs

Input	Dimension Type	Description
First	1-by-1	Contains the command for the signal that the pilot model controls.
Second	1-by-1	Contains the signal that the pilot model controls.

Output	Dimension Type	Description
First	1-by-1	Contains the command for the aircraft.

## References

[1] McRuer, D. T., Krendel, E., *Mathematical Models of Human Pilot Behavior*. Advisory Group on Aerospace Research and Development AGARDograph 180, Jan. 1974.

## See Also

Precision Pilot Model | Tustin Pilot Model |

# Custom Variable Mass 3DoF (Body Axes)

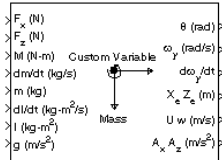
## Purpose

Implement three-degrees-of-freedom equations of motion of custom variable mass with respect to body axes

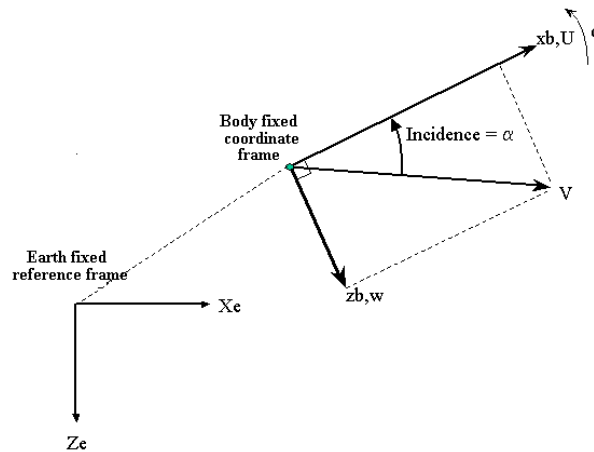
## Library

Equations of Motion/3DoF

## Description



The Custom Variable Mass 3DoF (Body Axes) block considers the rotation in the vertical plane of a body-fixed coordinate frame about an Earth-fixed reference frame.



The equations of motion are

$$\dot{u} = \frac{F_x}{m} - \frac{\dot{m}U}{m} - qw - g \sin \theta$$

$$\dot{w} = \frac{F_z}{m} - \frac{\dot{m}w}{m} + qu + g \cos \theta$$

$$\dot{q} = \frac{M - \dot{I}_{yy}q}{I_{yy}}$$

$$\dot{\theta} = q$$



# Custom Variable Mass 3DoF (Body Axes)

where the applied forces are assumed to act at the center of gravity of the body.

## Dialog Box

Function Block Parameters: Custom Variable Mass 3DoF (Body Axes)

3DoF EoM (mask) (link)

Integrate the three-degrees-of-freedom equations of motion to determine body position, velocity, attitude, and related values.

Parameters

Units: Metric (MKS)

Mass type: Custom Variable

Initial velocity:  
100

Initial body attitude:  
0

Initial incidence:  
0

Initial body rotation rate:  
0

Initial position (x z):  
[0 0]

Gravity source: External

OK Cancel Help Apply

### Units

Specifies the input and output units:

# Custom Variable Mass 3DoF (Body Axes)

---

Units	Forces	Moment	Acceleration	Velocity	Position	Mass	Inertia
Metric (MKS)	Newton	Newton meter	Meters per second squared	Meters per second	Meters	Kilogram	Kilogram meter squared
English (Velocity in ft/s)	Pound	Foot pound	Feet per second squared	Feet per second	Feet	Slug	Slug foot squared
English (Velocity in kts)	Pound	Foot pound	Feet per second squared	Knots	Feet	Slug	Slug foot squared

## Mass Type

Select the type of mass to use:

- Fixed                      Mass is constant throughout the simulation.
- Simple Variable        Mass and inertia vary linearly as a function of mass rate.
- Custom Variable        Mass and inertia variations are customizable.

The Custom Variable selection conforms to the previously described equations of motion.

## Initial velocity

A scalar value for the initial velocity of the body, ( $V_0$ ).

## Initial body attitude

A scalar value for the initial pitch attitude of the body, ( $\theta_0$ ).

## Initial incidence

A scalar value for the initial angle between the velocity vector and the body, ( $\alpha_0$ ).

# Custom Variable Mass 3DoF (Body Axes)

## Initial body rotation rate

A scalar value for the initial body rotation rate, ( $q_0$ ).

## Initial position (x,z)

A two-element vector containing the initial location of the body in the Earth-fixed reference frame.

## Gravity Source

Specify source of gravity:

External	Variable gravity input to block
Internal	Constant gravity specified in mask

## Acceleration due to gravity

A scalar value for the acceleration due to gravity used if internal gravity source is selected. If gravity is to be neglected in the simulation, this value can be set to 0.

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the force acting along the body $x$ -axis, ( $F_x$ ).
Second		Contains the force acting along the body $z$ -axis, ( $F_z$ ).
Third		Contains the applied pitch moment, ( $M$ ).
Fourth		Contains the rate of change of mass, ( $\dot{m}$ )
Fifth		Contains the mass, ( $m$ ).
Sixth		Contains the rate of change of inertia tensor matrix, ( $\dot{I}_{yy}$ ).
Seventh		Contains the inertia tensor matrix, ( $I_{yy}$ ).
Eighth (Optional)		Contains the gravity in the selected units.

## Custom Variable Mass 3DoF (Body Axes)

---

Output Dimension Type	Description
First	Contains the pitch attitude, in radians ( $\theta$ ).
Second	Contains the pitch angular rate, in radians per second ( $\dot{q}$ ).
Third	Contains the pitch angular acceleration, in radians per second squared ( $\ddot{q}$ ).
Fourth Two-element vector	Contains the location of the body, in the Earth-fixed reference frame, ( $X_e, Z_e$ ).
Fifth Two-element vector	Contains the velocity of the body resolved into the body-fixed coordinate frame, ( $u, w$ ).
Sixth	Contains the acceleration of the body resolved into the body-fixed coordinate frame, ( $A_x, A_z$ ).

### See Also

3DoF (Body Axes)

Incidence & Airspeed

Simple Variable Mass 3DoF (Body Axes)

# Custom Variable Mass 3DoF (Wind Axes)

## Purpose

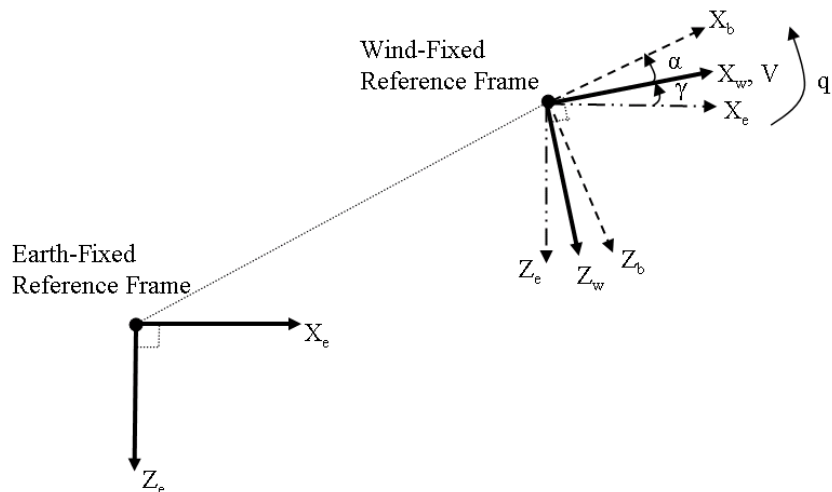
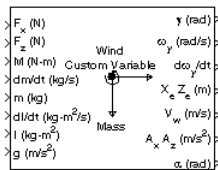
Implement three-degrees-of-freedom equations of motion of custom variable mass with respect to wind axes

## Library

Equations of Motion/3DoF

## Description

The Custom Variable Mass 3DoF (Wind Axes) block considers the rotation in the vertical plane of a wind-fixed coordinate frame about an Earth-fixed reference frame.



The equations of motion are

$$\begin{aligned}\dot{V} &= \frac{F_{x_{wind}}}{m} - \frac{\dot{m}V}{m} - g \sin \gamma \\ \dot{\alpha} &= \frac{F_{z_{wind}}}{mV} + q + \frac{g}{V} \cos \gamma \\ \dot{q} &= \dot{\theta} = \frac{M_{y_{body}} - \dot{I}_{yy}q}{I_{yy}} \\ \dot{\gamma} &= q - \dot{\alpha}\end{aligned}$$

# Custom Variable Mass 3DoF (Wind Axes)

where the applied forces are assumed to act at the center of gravity of the body.

## Dialog Box

Function Block Parameters: Custom Variable Mass 3DoF (Wind Axes)

3DoF Wind EoM (mask) (link)

Integrate the three-degrees-of-freedom equations of motion in wind axes to determine position, velocity, attitude, and related values.

Parameters

Units: Metric (MKS)

Mass type: Custom Variable

Initial airspeed: 100

Initial flight path angle: 0

Initial incidence: 0

Initial body rotation rate: 0

Initial position (x z): [0 0]

Gravity source: External

OK Cancel Help Apply

## Units

Specifies the input and output units:

# Custom Variable Mass 3DoF (Wind Axes)

Units	Forces	Moment	Acceleration	Velocity	Position	Mass	Inertia
Metric (MKS)	Newton	Newton meter	Meters per second squared	Meters per second	Meters	Kilogram	Kilogram meter squared
English (Velocity in ft/s)	Pound	Foot pound	Feet per second squared	Feet per second	Feet	Slug	Slug foot squared
English (Velocity in kts)	Pound	Foot pound	Feet per second squared	Knots	Feet	Slug	Slug foot squared

## Mass Type

Select the type of mass to use:

Fixed	Mass is constant throughout the simulation.
Simple Variable	Mass and inertia vary linearly as a function of mass rate.
Custom Variable	Mass and inertia variations are customizable.

The Custom Variable selection conforms to the previously described equations of motion.

## Initial airspeed

A scalar value for the initial velocity of the body, ( $V_0$ ).

## Initial flight path angle

A scalar value for the initial pitch attitude of the body, ( $\gamma_0$ ).

# Custom Variable Mass 3DoF (Wind Axes)

---

## Initial incidence

A scalar value for the initial angle between the velocity vector and the body, ( $\alpha_0$ ).

## Initial body rotation rate

A scalar value for the initial body rotation rate, ( $q_0$ ).

## Initial position (x,z)

A two-element vector containing the initial location of the body in the Earth-fixed reference frame.

## Gravity Source

Specify source of gravity:

External	Variable gravity input to block
Internal	Constant gravity specified in mask

## Acceleration due to gravity

A scalar value for the acceleration due to gravity used if internal gravity source is selected. If gravity is to be neglected in the simulation, this value can be set to 0.

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the force acting along the wind $x$ -axis, ( $F_x$ ).
Second		Contains the force acting along the wind $z$ -axis, ( $F_z$ ).
Third		Contains the applied pitch moment in body axes, ( $M$ ).
Fourth		Contains the rate of change of mass, ( $\dot{m}$ ).
Fifth		Contains the mass, ( $m$ ).



# Custom Variable Mass 3DoF (Wind Axes)

Input	Dimension Type	Description
Sixth		Contains the rate of change of inertia tensor matrix, $(\dot{I}_{yy})$ .
Seventh		Contains the inertia tensor matrix, $(I_{yy})$ .
Eighth (Optional)		Contains the gravity in the selected units.

Output	Dimension Type	Description
First		Contains the flight path angle, in radians ( $\gamma$ ).
Second		Contains the pitch angular rate, in radians per second ( $\omega_y$ ).
Third		Contains the pitch angular acceleration, in radians per second squared ( $d\omega_y/dt$ ).
Fourth	Two-element vector	Contains the location of the body, in the Earth-fixed reference frame, $(X_e, Z_e)$ .
Fifth	Two-element vector	Contains the velocity of the body resolved into the wind-fixed coordinate frame, $(V, 0)$ .
Sixth	Two-element vector	Contains the acceleration of the body resolved into the body-fixed coordinate frame, $(A_x, A_z)$ .
Seventh	Scalar	Contains the angle of attack, $(\alpha)$ .

## Reference

Stevens, B. L., and F. L. Lewis, *Aircraft Control and Simulation*, John Wiley & Sons, New York, 1992.

## See Also

3DoF (Body Axes)

3DoF (Wind Axes)

4th Order Point Mass (Longitudinal)

# Custom Variable Mass 3DoF (Wind Axes)

---

Custom Variable Mass 3DoF (Body Axes)

Simple Variable Mass 3DoF (Body Axes)

Simple Variable Mass 3DoF (Wind Axes)

# Custom Variable Mass 6DoF (Euler Angles)

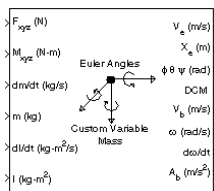
## Purpose

Implement Euler angle representation of six-degrees-of-freedom equations of motion of custom variable mass

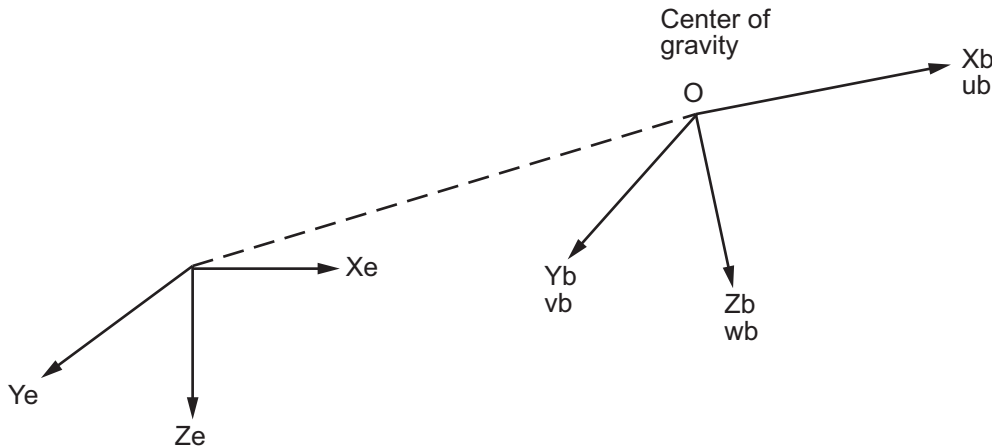
## Library

Equations of Motion/6DoF

## Description



The Custom Variable Mass 6DoF (Euler Angles) block considers the rotation of a body-fixed coordinate frame ( $X_b, Y_b, Z_b$ ) about a flat Earth reference frame ( $X_e, Y_e, Z_e$ ). The origin of the body-fixed coordinate frame is the center of gravity of the body, and the body is assumed to be rigid, an assumption that eliminates the need to consider the forces acting between individual elements of mass. The flat Earth reference frame is considered inertial, an excellent approximation that allows the forces due to the Earth's motion relative to the "fixed stars" to be neglected.



Flat Earth reference frame

The translational motion of the body-fixed coordinate frame is given below, where the applied forces  $[F_x \ F_y \ F_z]^T$  are in the body-fixed frame.

# Custom Variable Mass 6DoF (Euler Angles)

---

$$\bar{\mathbf{F}}_b = \begin{bmatrix} \mathbf{F}_x \\ \mathbf{F}_y \\ \mathbf{F}_z \end{bmatrix} = m(\dot{\bar{\mathbf{V}}}_b + \bar{\boldsymbol{\omega}} \times \bar{\mathbf{V}}_b) + \dot{m}\bar{\mathbf{V}}_b$$

$$\bar{\mathbf{V}}_b = \begin{bmatrix} u_b \\ v_b \\ w_b \end{bmatrix}, \bar{\boldsymbol{\omega}} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

The rotational dynamics of the body-fixed frame are given below, where the applied moments are  $[L \ M \ N]^T$ , and the inertia tensor  $I$  is with respect to the origin O.

$$\bar{\mathbf{M}}_B = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = I\dot{\bar{\boldsymbol{\omega}}} + \bar{\boldsymbol{\omega}} \times (I\bar{\boldsymbol{\omega}}) + \dot{I}\bar{\boldsymbol{\omega}}$$

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix}$$

$$\dot{I} = \begin{bmatrix} \dot{I}_{xx} & -\dot{I}_{xy} & -\dot{I}_{xz} \\ -\dot{I}_{yx} & \dot{I}_{yy} & -\dot{I}_{yz} \\ -\dot{I}_{zx} & -\dot{I}_{zy} & \dot{I}_{zz} \end{bmatrix}$$

The relationship between the body-fixed angular velocity vector,  $[p \ q \ r]^T$ , and the rate of change of the Euler angles,  $[\dot{\phi} \ \dot{\theta} \ \dot{\psi}]^T$ , can be determined by resolving the Euler rates into the body-fixed coordinate frame.

## Custom Variable Mass 6DoF (Euler Angles)

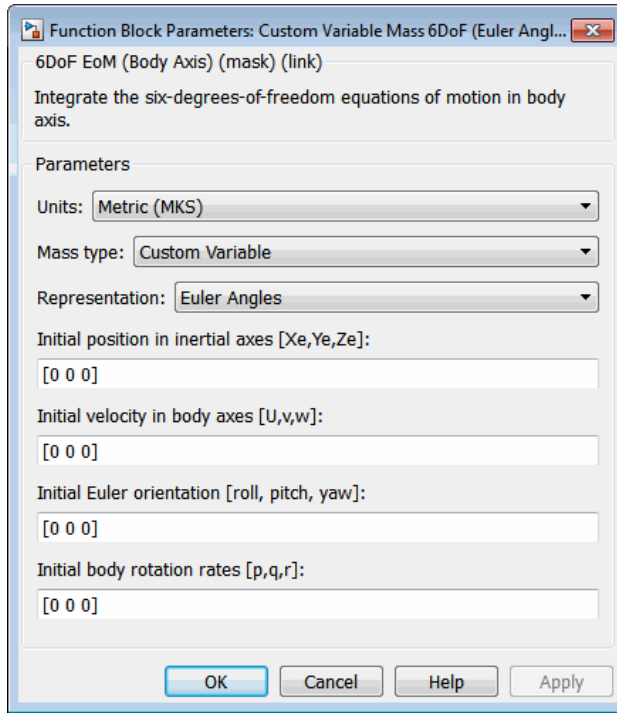
$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} = J^{-1} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

Inverting  $J$  then gives the required relationship to determine the Euler rate vector.

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = J \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & (\sin\phi \tan\theta) & (\cos\phi \tan\theta) \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

# Custom Variable Mass 6DoF (Euler Angles)

## Dialog Box



## Units

Specifies the input and output units:

# Custom Variable Mass 6DoF (Euler Angles)

Units	Forces	Moment	Acceleration	Velocity	Position	Mass	Inertia
Metric (MKS)	Newton	Newton meter	Meters per second squared	Meters per second	Meters	Kilogram	Kilogram meter squared
English (Velocity in ft/s)	Pound	Foot pound	Feet per second squared	Feet per second	Feet	Slug	Slug foot squared
English (Velocity in kts)	Pound	Foot pound	Feet per second squared	Knots	Feet	Slug	Slug foot squared

## Mass Type

Select the type of mass to use:

Fixed	Mass is constant throughout the simulation.
Simple Variable	Mass and inertia vary linearly as a function of mass rate.
Custom Variable	Mass and inertia variations are customizable.

The Custom Variable selection conforms to the previously described equations of motion.

## Representation

Select the representation to use:

Euler Angles	Use Euler angles within equations of motion.
Quaternion	Use quaternions within equations of motion.

# Custom Variable Mass 6DoF (Euler Angles)

---

The Euler Angles selection conforms to the previously described equations of motion.

## **Initial position in inertial axes**

The three-element vector for the initial location of the body in the flat Earth reference frame.

## **Initial velocity in body axes**

The three-element vector for the initial velocity in the body-fixed coordinate frame.

## **Initial Euler rotation**

The three-element vector for the initial Euler rotation angles [roll, pitch, yaw], in radians.

## **Initial body rotation rates**

The three-element vector for the initial body-fixed angular rates, in radians per second.

## **Inputs and Outputs**

<b>Input</b>	<b>Dimension Type</b>	<b>Description</b>
First	Vector	Contains the three applied forces.
Second	Vector	Contains the three applied moments.
Third	Scalar	Contains the rate of change of mass.
Fourth	Scalar	Contains the mass.
Fifth	3-by-3 matrix	Contains the rate of change of inertia tensor matrix.
Sixth	3-by-3 matrix	Contains the inertia tensor matrix.



# Custom Variable Mass 6DoF (Euler Angles)

Output Dimension	Type	Description
First	Three-element vector	Contains the velocity in the flat Earth reference frame.
Second	Three-element vector	Contains the position in the flat Earth reference frame.
Third	Three-element vector	Contains the Euler rotation angles [roll, pitch, yaw], in radians.
Fourth	3-by-3 matrix	Contains the coordinate transformation from flat Earth axes to body-fixed axes.
Fifth	Three-element vector	Contains the velocity in the body-fixed frame.
Sixth	Three-element vector	Contains the angular rates in body-fixed axes, in radians per second.
Seventh	Three-element vector	Contains the angular accelerations in body-fixed axes, in radians per second squared.
Eight	Three-element vector	Contains the accelerations in body-fixed axes.

## Assumptions and Limitations

The block assumes that the applied forces are acting at the center of gravity of the body.

## Reference

Mangiacasale, L., *Flight Mechanics of a  $\mu$ -Airplane with a MATLAB Simulink Helper*, Edizioni Libreria CLUP, Milan, 1998.

## See Also

6DoF (Euler Angles)  
6DoF (Quaternion)  
6DoF ECEF (Quaternion)

## Custom Variable Mass 6DoF (Euler Angles)

---

6DoF Wind (Quaternion)  
6DoF Wind (Wind Angles)  
6th Order Point Mass (Coordinated Flight)  
Custom Variable Mass 6DoF (Quaternion)  
Custom Variable Mass 6DoF ECEF (Quaternion)  
Custom Variable Mass 6DoF Wind (Quaternion)  
Custom Variable Mass 6DoF Wind (Wind Angles)  
Simple Variable Mass 6DoF (Euler Angles)  
Simple Variable Mass 6DoF (Quaternion)  
Simple Variable Mass 6DoF ECEF (Quaternion)  
Simple Variable Mass 6DoF Wind (Quaternion)  
Simple Variable Mass 6DoF Wind (Wind Angles)

# Custom Variable Mass 6DoF (Quaternion)

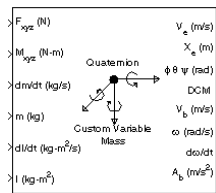
## Purpose

Implement quaternion representation of six-degrees-of-freedom equations of motion of custom variable mass with respect to body axes

## Library

Equations of Motion/6DoF

## Description



For a description of the coordinate system and the translational dynamics, see the block description for the Custom Variable Mass 6DoF (Euler Angles) block.

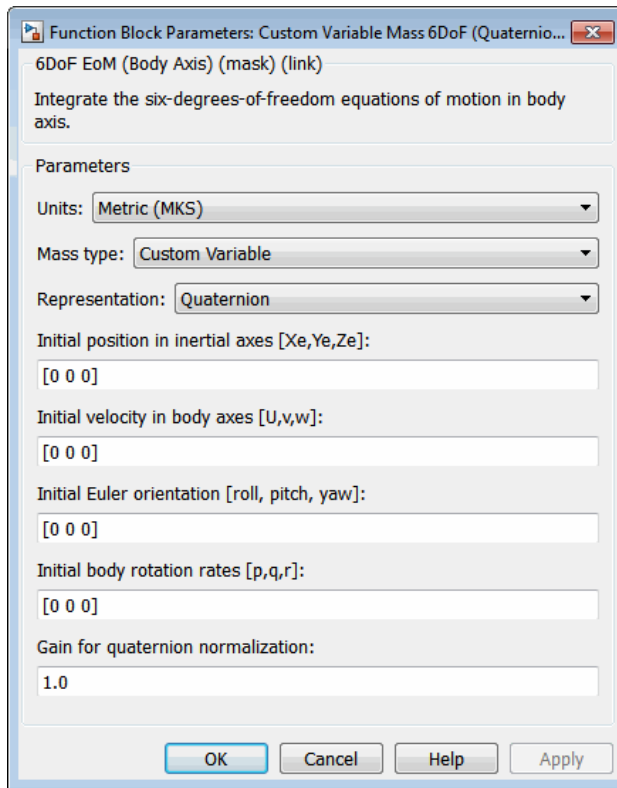
The integration of the rate of change of the quaternion vector is given below. The gain  $K$  drives the norm of the quaternion state vector to 1.0 should  $\varepsilon$  become nonzero. You must choose the value of this gain with care, because a large value improves the decay rate of the error in the norm, but also slows the simulation because fast dynamics are introduced. An error in the magnitude in one element of the quaternion vector is spread equally among all the elements, potentially increasing the error in the state vector.

$$\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} + K\varepsilon \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

$$\varepsilon = 1 - (q_0^2 + q_1^2 + q_2^2 + q_3^2)$$

# Custom Variable Mass 6DoF (Quaternion)

## Dialog Box



### Units

Specifies the input and output units:

# Custom Variable Mass 6DoF (Quaternion)

Units	Forces	Moment	Acceleration	Velocity	Position	Mass	Inertia
Metric (MKS)	Newton	Newton meter	Meters per second squared	Meters per second	Meters	Kilogram	Kilogram meter squared
English (Velocity in ft/s)	Pound	Foot pound	Feet per second squared	Feet per second	Feet	Slug	Slug foot squared
English (Velocity in kts)	Pound	Foot pound	Feet per second squared	Knots	Feet	Slug	Slug foot squared

## Mass Type

Select the type of mass to use:

- Fixed                      Mass is constant throughout the simulation.
- Simple Variable        Mass and inertia vary linearly as a function of mass rate.
- Custom Variable        Mass and inertia variations are customizable.

The Custom Variable selection conforms to the previously described equations of motion.

## Representation

Select the representation to use:

- Euler Angles            Use Euler angles within equations of motion.
- Quaternion              Use quaternions within equations of motion.

# Custom Variable Mass 6DoF (Quaternion)

---

The Quaternion selection conforms to the previously described equations of motion.

## **Initial position in inertial axes**

The three-element vector for the initial location of the body in the flat Earth reference frame.

## **Initial velocity in body axes**

The three-element vector for the initial velocity in the body-fixed coordinate frame.

## **Initial Euler rotation**

The three-element vector for the initial Euler rotation angles [roll, pitch, yaw], in radians.

## **Initial body rotation rates**

The three-element vector for the initial body-fixed angular rates, in radians per second.

## **Gain for quaternion normalization**

The gain to maintain the norm of the quaternion vector equal to 1.0.

## **Inputs and Outputs**

<b>Input</b>	<b>Dimension Type</b>	<b>Description</b>
First	Vector	Contains the three applied forces.
Second	Vector	Contains the three applied moments.
Third	Scalar	Contains the rate of change of mass.
Fourth	Scalar	Contains the mass.
Fifth	3-by-3 matrix	Contains the rate of change of inertia tensor matrix.
Sixth	3-by-3 matrix	Contains the inertia tensor matrix.

# Custom Variable Mass 6DoF (Quaternion)

Output Dimension	Type	Description
First	Three-element vector	Contains the velocity in the flat Earth reference frame.
Second	Three-element vector	Contains the position in the flat Earth reference frame.
Third	Three-element vector	Contains the Euler rotation angles [roll, pitch, yaw], in radians.
Fourth	3-by-3 matrix	Contains the coordinate transformation from flat Earth axes to body-fixed axes.
Fifth	Three-element vector	Contains the velocity in the body-fixed frame.
Sixth	Three-element vector	Contains the angular rates in body-fixed axes, in radians per second.
Seventh	Three-element vector	Contains the angular accelerations in body-fixed axes, in radians per second squared.
Eight	Three-element vector	Contains the accelerations in body-fixed axes.

## Assumptions and Limitations

The block assumes that the applied forces are acting at the center of gravity of the body.

## Reference

Mangiacasale, L., *Flight Mechanics of a u-Airplane with a MATLAB Simulink Helper*, Edizioni Libreria CLUP, Milan, 1998.

## See Also

6DoF (Euler Angles)  
6DoF (Quaternion)  
6DoF ECEF (Quaternion)

## Custom Variable Mass 6DoF (Quaternion)

---

6DoF Wind (Quaternion)

6DoF Wind (Wind Angles)

6th Order Point Mass (Coordinated Flight)

Custom Variable Mass 6DoF (Euler Angles)

Custom Variable Mass 6DoF ECEF (Quaternion)

Custom Variable Mass 6DoF Wind (Quaternion)

Custom Variable Mass 6DoF Wind (Wind Angles)

Simple Variable Mass 6DoF (Euler Angles)

Simple Variable Mass 6DoF (Quaternion)

Simple Variable Mass 6DoF ECEF (Quaternion)

Simple Variable Mass 6DoF Wind (Quaternion)

Simple Variable Mass 6DoF Wind (Wind Angles)



# Custom Variable Mass 6DoF ECEF (Quaternion)

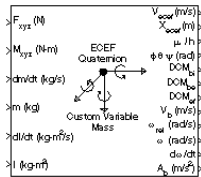
## Purpose

Implement quaternion representation of six-degrees-of-freedom equations of motion of custom variable mass in Earth-centered Earth-fixed (ECEF) coordinates

## Library

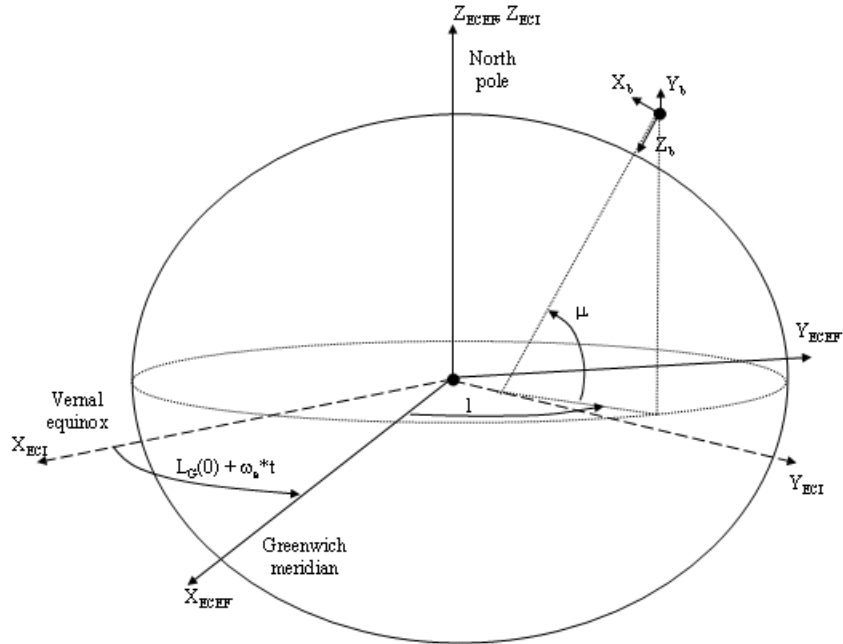
Equations of Motion/6DoF

## Description



The Custom Variable Mass 6DoF ECEF (Quaternion) block considers the rotation of a Earth-centered Earth-fixed (ECEF) coordinate frame ( $X_{ECEF}$ ,  $Y_{ECEF}$ ,  $Z_{ECEF}$ ) about an Earth-centered inertial (ECI) reference frame ( $X_{ECI}$ ,  $Y_{ECI}$ ,  $Z_{ECI}$ ). The origin of the ECEF coordinate frame is the center of the Earth, additionally the body of interest is assumed to be rigid, an assumption that eliminates the need to consider the forces acting between individual elements of mass. The representation of the rotation of ECEF frame from ECI frame is simplified to consider only the constant rotation of the ellipsoid Earth ( $\omega_e$ ) including an initial celestial longitude ( $L_G(0)$ ). This excellent approximation allows the forces due to the Earth's complex motion relative to the "fixed stars" to be neglected.

# Custom Variable Mass 6DoF ECEF (Quaternion)



The translational motion of the ECEF coordinate frame is given below, where the applied forces  $[F_x \ F_y \ F_z]^T$  are in the body frame.

$$\begin{aligned} \bar{\mathbf{F}}_b = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} &= m \left( \dot{\bar{\mathbf{V}}}_b + \bar{\omega}_b \times \bar{\mathbf{V}}_b + DCM_{bf} \bar{\omega}_e \times \bar{\mathbf{V}}_b \right) + DCM_{bf} \left( \bar{\omega}_e \times (\bar{\omega}_e \times \bar{\mathbf{X}}_f) \right) \\ &+ \dot{m} \left( \bar{\mathbf{V}}_b + DCM_{bf} (\bar{\omega}_e \times \bar{\mathbf{X}}_f) \right) \end{aligned}$$

where the change of position in ECEF  $\dot{\bar{\mathbf{x}}}_f$  is calculated by

$$\dot{\bar{\mathbf{x}}}_f = DCM_{fb} \bar{\mathbf{V}}_b$$

# Custom Variable Mass 6DoF ECEF (Quaternion)

and the velocity of the body with respect to ECEF frame, expressed in body frame ( $\bar{V}_b$ ), angular rates of the body with respect to ECI frame, expressed in body frame ( $\bar{\omega}_b$ ). Earth rotation rate ( $\bar{\omega}_e$ ), and relative angular rates of the body with respect to north-east-down (NED) frame, expressed in body frame ( $\bar{\omega}_{rel}$ ) are defined as

$$\bar{V}_b = \begin{bmatrix} u \\ v \\ w \end{bmatrix}, \bar{\omega}_{rel} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}, \bar{\omega}_e = \begin{bmatrix} 0 \\ 0 \\ \omega_e \end{bmatrix}, \bar{\omega}_b = \bar{\omega}_{rel} + DCM_{bf} \bar{\omega}_e + DCM_{be} \bar{\omega}_{ned}$$

$$\bar{\omega}_{ned} = \begin{bmatrix} \dot{l} \cos \mu \\ -\dot{\mu} \\ -\dot{l} \sin \mu \end{bmatrix} = \begin{bmatrix} V_E / (N + h) \\ -V_N / (M + h) \\ V_E \bullet \tan \mu / (N + h) \end{bmatrix}$$

The rotational dynamics of the body defined in body-fixed frame are given below, where the applied moments are  $[L \ M \ N]^T$ , and the inertia tensor  $I$  is with respect to the origin O.

$$\bar{M}_b = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = \bar{I} \dot{\bar{\omega}}_b + \bar{\omega}_b \times (\bar{I} \bar{\omega}_b) + \dot{I} \bar{\omega}_b$$

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix}$$

The rate of change of the inertia tensor is defined by the following equation.

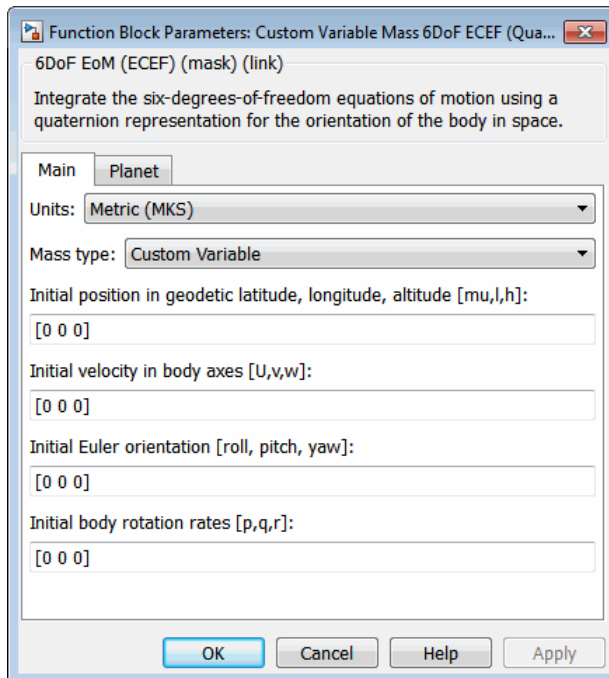
# Custom Variable Mass 6DoF ECEF (Quaternion)

$$\dot{I} = \begin{bmatrix} \dot{I}_{xx} & -\dot{I}_{xy} & -\dot{I}_{xz} \\ -\dot{I}_{yx} & \dot{I}_{yy} & -\dot{I}_{yz} \\ -\dot{I}_{zx} & -\dot{I}_{zy} & \dot{I}_{zz} \end{bmatrix}$$

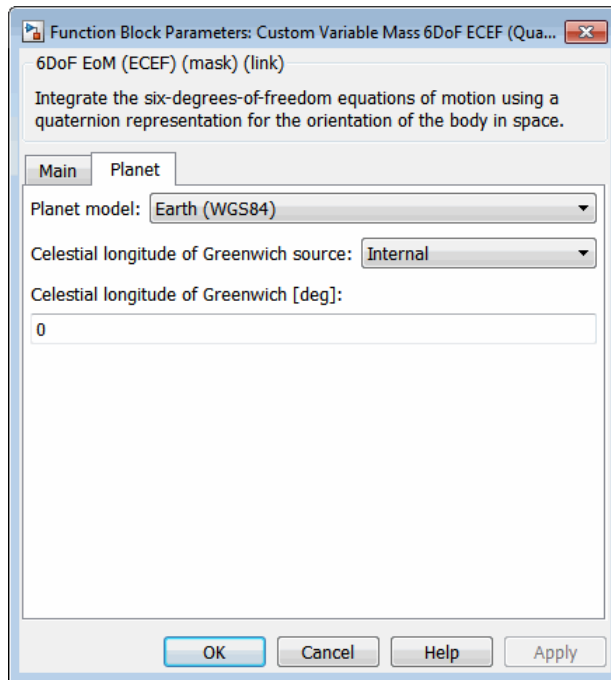
The integration of the rate of change of the quaternion vector is given below.

$$\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = -\frac{1}{2} \begin{bmatrix} 0 & \omega_b(1) & \omega_b(2) & \omega_b(3) \\ -\omega_b(1) & 0 & -\omega_b(3) & \omega_b(2) \\ -\omega_b(2) & \omega_b(3) & 0 & -\omega_b(1) \\ -\omega_b(3) & -\omega_b(2) & \omega_b(1) & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

## Dialog Box



# Custom Variable Mass 6DoF ECEF (Quaternion)



## Units

Specifies the input and output units:

# Custom Variable Mass 6DoF ECEF (Quaternion)

---

<b>Units</b>	<b>Forces</b>	<b>Moment</b>	<b>Acceleration</b>	<b>Velocity</b>	<b>Position</b>	<b>Mass</b>	<b>Inertia</b>
Metric (MKS)	Newton	Newton meter	Meters per second squared	Meters per second	Meters	Kilogram	Kilogram meter squared
English (Velocity in ft/s)	Pound	Foot pound	Feet per second squared	Feet per second	Feet	Slug	Slug foot squared
English (Velocity in kts)	Pound	Foot pound	Feet per second squared	Knots	Feet	Slug	Slug foot squared

## Mass type

Select the type of mass to use:

Fixed	Mass is constant throughout the simulation (see 6DoF ECEF (Quaternion)).
Simple Variable	Mass and inertia vary linearly as a function of mass rate (see Simple Variable Mass 6DoF ECEF (Quaternion)).
Custom Variable	Mass and inertia variations are customizable.

The Simple Variable selection conforms to the previously described equations of motion.

## Initial position in geodetic latitude, longitude and altitude

The three-element vector for the initial location of the body in the geodetic reference frame.

## Initial velocity in body-axis

The three-element vector containing the initial velocity of the body with respect to ECEF frame, expressed in body frame.

# Custom Variable Mass 6DoF ECEF (Quaternion)

---

## **Initial Euler orientation**

The three-element vector containing the initial Euler rotation angles [roll, pitch, yaw], in radians. Euler rotation angles are those between the body and north-east-down (NED) coordinate systems.

## **Initial body rotation rates**

The three-element vector for the initial angular rates of the body with respect to NED frame, expressed in body frame, in radians per second.

## **Planet model**

Specifies the planet model to use, Custom or Earth (WGS84).

## **Flattening**

Specifies the flattening of the planet. This option is only available when **Planet model** is set to Custom.

## **Equatorial radius of planet**

Specifies the radius of the planet at its equator. The units of the equatorial radius parameter should be the same as the units for ECEF position. This option is only available when **Planet model** is set to Custom.

## **Rotational rate**

Specifies the scalar rotational rate of the planet in rad/s. This option is only available when **Planet model** is set to Custom.

## **Celestial longitude of Greenwich source**

Specifies the source of Greenwich meridian's initial celestial longitude:

Internal	Use celestial longitude value from mask dialog.
External	Use external input for celestial longitude value.

# Custom Variable Mass 6DoF ECEF (Quaternion)

## Celestial longitude of Greenwich

The initial angle between Greenwich meridian and the  $x$ -axis of the ECI frame.

### Inputs and Outputs

Input	Dimension Type	Description
First	Vector	Contains the three applied forces in body-fixed axes.
Second	Vector	Contains the three applied moments in body-fixed axes.
Third	Scalar	Contains the rate of change of mass.
Fourth	Scalar	Contains the mass.
Fifth	3-by-3 matrix	Applies to the rate of change of inertia tensor matrix.
Sixth	3-by-3 matrix	Applies to the inertia tensor matrix.

Output	Dimension Type	Description
First	Three-element vector	Contains the velocity of the body with respect to ECEF frame, expressed in ECEF frame.
Second	Three-element vector	Contains the position in the ECEF reference frame.
Third	Three-element vector	Contains the position in geodetic latitude, longitude and altitude, in degrees, degrees and selected units of length respectively.
Fourth	Three-element vector	Contains the body rotation angles [roll, pitch, yaw], in radians. Euler rotation angles are those between the body and NED coordinate systems.



## Custom Variable Mass 6DoF ECEF (Quaternion)

Output	Dimension Type	Description
Fifth	3-by-3 matrix	Applies to the coordinate transformation from ECI axes to body-fixed axes.
Sixth	3-by-3 matrix	Applies to the coordinate transformation from NED axes to body-fixed axes.
Seventh	3-by-3 matrix	Applies to the coordinate transformation from ECEF axes to NED axes.
Eighth	Three-element vector	Contains the velocity of the body with respect to ECEF frame, expressed in body frame.
Ninth	Three-element vector	Contains the relative angular rates of the body with respect to NED frame, expressed in body frame, in radians per second.
Tenth	Three-element vector	Contains the angular rates of the body with respect to ECI frame, expressed in body frame, in radians per second.
Eleventh	Three-element vector	Contains the angular accelerations of the body with respect to ECI frame, expressed in body frame, in radians per second squared.
Twelfth	Three-element vector	Contains the accelerations in body-fixed axes.

# Custom Variable Mass 6DoF ECEF (Quaternion)

---

## Assumptions and Limitations

This implementation assumes that the applied forces are acting at the center of gravity of the body.

This implementation generates a geodetic latitude that lies between  $\pm 90$  degrees, and longitude that lies between  $\pm 180$  degrees. Additionally, the MSL altitude is approximate.

The Earth is assumed to be ellipsoidal. By setting flattening to 0.0, a spherical planet can be achieved. The Earth's precession, nutation, and polar motion are neglected. The celestial longitude of Greenwich is Greenwich Mean Sidereal Time (GMST) and provides a rough approximation to the sidereal time.

The implementation of the ECEF coordinate system assumes that the origin is at the center of the planet, the  $x$ -axis intersects the Greenwich meridian and the equator, the  $z$ -axis is the mean spin axis of the planet, positive to the north, and the  $y$ -axis completes the right-handed system.

The implementation of the ECI coordinate system assumes that the origin is at the center of the planet, the  $x$ -axis is the continuation of the line from the center of the Earth through the center of the Sun toward the vernal equinox, the  $z$ -axis points in the direction of the mean equatorial plane's north pole, positive to the north, and the  $y$ -axis completes the right-handed system.

## References

Stevens, B. L., and F. L. Lewis, *Aircraft Control and Simulation, Second Edition*, John Wiley & Sons, New York, 2003.

McFarland, Richard E., *A Standard Kinematic Model for Flight simulation at NASA-Ames*, NASA CR-2497.

"Supplement to Department of Defense World Geodetic System 1984 Technical Report: Part I - Methods, Techniques and Data Used in WGS84 Development," DMA TR8350.2-A.

## See Also

6DoF (Euler Angles)

6DoF (Quaternion)

6DoF ECEF (Quaternion)

# Custom Variable Mass 6DoF ECEF (Quaternion)

---

6DoF Wind (Quaternion)

6DoF Wind (Wind Angles)

6th Order Point Mass (Coordinated Flight)

Custom Variable Mass 6DoF (Euler Angles)

Custom Variable Mass 6DoF (Quaternion)

Custom Variable Mass 6DoF Wind (Quaternion)

Custom Variable Mass 6DoF Wind (Wind Angles)

Simple Variable Mass 6DoF (Euler Angles)

Simple Variable Mass 6DoF (Quaternion)

Simple Variable Mass 6DoF ECEF (Quaternion)

Simple Variable Mass 6DoF Wind (Quaternion)

Simple Variable Mass 6DoF Wind (Wind Angles)

# Custom Variable Mass 6DoF Wind (Quaternion)

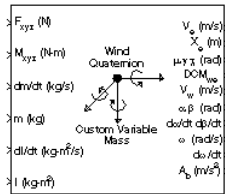
## Purpose

Implement quaternion representation of six-degrees-of-freedom equations of motion of custom variable mass with respect to wind axes

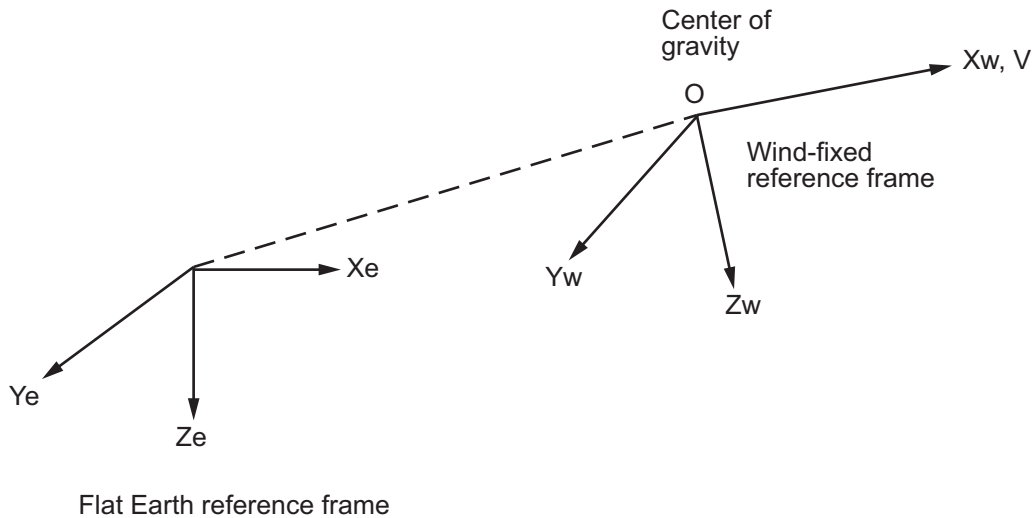
## Library

Equations of Motion/6DoF

## Description



The Custom Variable Mass 6DoF Wind (Quaternion) block considers the rotation of a wind-fixed coordinate frame ( $X_w, Y_w, Z_w$ ) about an flat Earth reference frame ( $X_e, Y_e, Z_e$ ). The origin of the wind-fixed coordinate frame is the center of gravity of the body, and the body is assumed to be rigid, an assumption that eliminates the need to consider the forces acting between individual elements of mass. The flat Earth reference frame is considered inertial, an excellent approximation that allows the forces due to the Earth's motion relative to the "fixed stars" to be neglected.



The translational motion of the wind-fixed coordinate frame is given below, where the applied forces  $[F_x, F_y, F_z]^T$  are in the wind-fixed frame.

# Custom Variable Mass 6DoF Wind (Quaternion)

$$\bar{F}_w = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = m(\dot{\bar{V}}_w + \bar{\omega}_w \times \bar{V}_w) + \dot{m}\bar{V}_w$$

$$\bar{V}_w = \begin{bmatrix} V \\ 0 \\ 0 \end{bmatrix}, \bar{\omega}_w = \begin{bmatrix} p_w \\ q_w \\ r_w \end{bmatrix} = DMC_{wb} \begin{bmatrix} p_b - \dot{\beta} \sin \alpha \\ q_b - \dot{\alpha} \\ r_b + \dot{\beta} \cos \alpha \end{bmatrix}, \bar{\omega}_b = \begin{bmatrix} p_b \\ q_b \\ r_b \end{bmatrix}$$

The rotational dynamics of the body-fixed frame are given below, where the applied moments are  $[L \ M \ N]^T$ , and the inertia tensor  $I$  is with respect to the origin O. Inertia tensor  $I$  is much easier to define in body-fixed frame.

$$\bar{M}_b = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = I\dot{\bar{\omega}}_b + \bar{\omega}_b \times (I\bar{\omega}_b) + \dot{I}\bar{\omega}_b$$

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix}$$

The integration of the rate of change of the quaternion vector is given below.

$$\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = -\frac{1}{2} \begin{bmatrix} 0 & p & q & r \\ -p & 0 & -r & q \\ -q & r & 0 & -p \\ -r & -q & p & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

# Custom Variable Mass 6DoF Wind (Quaternion)

## Dialog Box

Function Block Parameters: Custom Variable Mass 6DoF Wind (Qua... [X]

6DoF EoM (Wind Axis) (mask) (link)

Integrate the six-degrees-of-freedom equations of motion in wind axis.

Parameters

Units: Metric (MKS) [v]

Mass type: Custom Variable [v]

Representation: Quaternion [v]

Initial position in inertial axes [Xe,Ye,Ze]:  
[0 0 0]

Initial airspeed, angle of attack, and sideslip angle [V,alpha,beta]:  
[0 0 0]

Initial wind orientation [bank angle,flight path angle,heading angle]:  
[0 0 0]

Initial body rotation rates [p,q,r]:  
[0 0 0]

OK Cancel Help Apply

### Units

Specifies the input and output units:

# Custom Variable Mass 6DoF Wind (Quaternion)

<b>Units</b>	<b>Forces</b>	<b>Moment</b>	<b>Acceleration</b>	<b>Velocity</b>	<b>Position</b>	<b>Mass</b>	<b>Inertia</b>
Metric (MKS)	Newton	Newton meter	Meters per second squared	Meters per second	Meters	Kilogram	Kilogram meter squared
English (Velocity in ft/s)	Pound	Foot pound	Feet per second squared	Feet per second	Feet	Slug	Slug foot squared
English (Velocity in kts)	Pound	Foot pound	Feet per second squared	Knots	Feet	Slug	Slug foot squared

## Mass Type

Select the type of mass to use:

Fixed	Mass is constant throughout the simulation.
Simple Variable	Mass and inertia vary linearly as a function of mass rate.
Custom Variable	Mass and inertia variations are customizable.

The Custom Variable selection conforms to the previously described equations of motion.

## Representation

Select the representation to use:

Wind Angles	Use wind angles within equations of motion.
Quaternion	Use quaternions within equations of motion.

# Custom Variable Mass 6DoF Wind (Quaternion)

---

The Quaternion selection conforms to the previously described equations of motion.

## **Initial position in inertial axes**

The three-element vector for the initial location of the body in the flat Earth reference frame.

## **Initial airspeed, sideslip angle, and angle of attack**

The three-element vector containing the initial airspeed, initial sideslip angle and initial angle of attack.

## **Initial wind orientation**

The three-element vector containing the initial wind angles [bank, flight path, and heading], in radians.

## **Initial body rotation rates**

The three-element vector for the initial body-fixed angular rates, in radians per second.

## **Inputs and Outputs**

<b>Input</b>	<b>Dimension Type</b>	<b>Description</b>
First	Vector	Contains the three applied forces in wind-fixed axes.
Second	Vector	Contains the three applied moments in body-fixed axes.
Third	Scalar	Contains the rate of change of mass.
Fourth	Scalar	Contains the mass.
Fifth	3-by-3 matrix	Applies to the rate of change of inertia tensor matrix in body-fixed axes.
Sixth	3-by-3 matrix	Applies to the inertia tensor matrix in body-fixed axes.



# Custom Variable Mass 6DoF Wind (Quaternion)

Output	Dimension Type	Description
First	Three-element vector	Contains the velocity in the flat Earth reference frame
Second	Three-element vector	Contains the position in the flat Earth reference frame.
Third	Three-element vector	Contains the wind rotation angles [bank, flight path, heading], in radians.
Fourth	3-by-3 matrix	Applies to the coordinate transformation from flat Earth axes to wind-fixed axes.
Fifth	Three-element vector	Contains to the velocity in the wind-fixed frame.
Sixth	Two-element vector	Contains the angle of attack and sideslip angle, in radians.
Seventh	Two-element vector	Contains the rate of change of angle of attack and rate of change of sideslip angle, in radians per second.
Eighth	Three-element vector	Contains the angular rates in body-fixed axes, in radians per second.
Ninth	Three-element vector	Contains the angular accelerations in body-fixed axes, in radians per second squared.
Tenth	Three-element vector	Contains the accelerations in body-fixed axes.

## Assumptions and Limitations

The block assumes that the applied forces are acting at the center of gravity of the body.

# Custom Variable Mass 6DoF Wind (Quaternion)

---

## References

Mangiacasale, L., *Flight Mechanics of a u-Airplane with a MATLAB Simulink Helper*, Edizioni Libreria CLUP, Milan, 1998.

Stevens, B. L., and F. L. Lewis, *Aircraft Control and Simulation*, John Wiley & Sons, New York, 1992.

## See Also

6DoF (Euler Angles)

6DoF (Quaternion)

6DoF ECEF (Quaternion)

6DoF Wind (Quaternion)

6DoF Wind (Wind Angles)

6th Order Point Mass (Coordinated Flight)

Custom Variable Mass 6DoF (Euler Angles)

Custom Variable Mass 6DoF (Quaternion)

Custom Variable Mass 6DoF ECEF (Quaternion)

Custom Variable Mass 6DoF Wind (Wind Angles)

Simple Variable Mass 6DoF (Euler Angles)

Simple Variable Mass 6DoF (Quaternion)

Simple Variable Mass 6DoF ECEF (Quaternion)

Simple Variable Mass 6DoF Wind (Quaternion)

Simple Variable Mass 6DoF Wind (Wind Angles)

# Custom Variable Mass 6DoF Wind (Wind Angles)

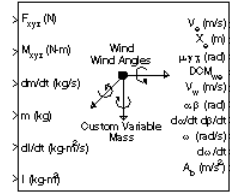
## Purpose

Implement wind angle representation of six-degrees-of-freedom equations of motion of custom variable mass

## Library

Equations of Motion/6DoF

## Description



For a description of the coordinate system employed and the translational dynamics, see the block description for the Custom Variable Mass 6DoF Wind (Quaternion) block.

The relationship between the wind angles,  $[\mu \ \gamma \ \chi]^T$ , can be determined by resolving the wind rates into the wind-fixed coordinate frame.

$$\begin{bmatrix} p_w \\ q_w \\ r_w \end{bmatrix} = \begin{bmatrix} \dot{\mu} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \mu & \sin \mu \\ 0 & -\sin \mu & \cos \mu \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\gamma} \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \mu & \sin \mu \\ 0 & -\sin \mu & \cos \mu \end{bmatrix} \begin{bmatrix} \cos \gamma & 0 & -\sin \gamma \\ 0 & 1 & 0 \\ \sin \gamma & 0 & \cos \gamma \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\chi} \end{bmatrix} \equiv J^{-1} \begin{bmatrix} \dot{\mu} \\ \dot{\gamma} \\ \dot{\chi} \end{bmatrix}$$

Inverting  $J$  then gives the required relationship to determine the wind rate vector.

$$\begin{bmatrix} \dot{\mu} \\ \dot{\gamma} \\ \dot{\chi} \end{bmatrix} = J \begin{bmatrix} p_w \\ q_w \\ r_w \end{bmatrix} = \begin{bmatrix} 1 & (\sin \mu \tan \gamma) & (\cos \mu \tan \gamma) \\ 0 & \cos \mu & -\sin \mu \\ 0 & \frac{\sin \mu}{\cos \gamma} & \frac{\cos \mu}{\cos \gamma} \end{bmatrix} \begin{bmatrix} p_w \\ q_w \\ r_w \end{bmatrix}$$

The body-fixed angular rates are related to the wind-fixed angular rate by the following equation.

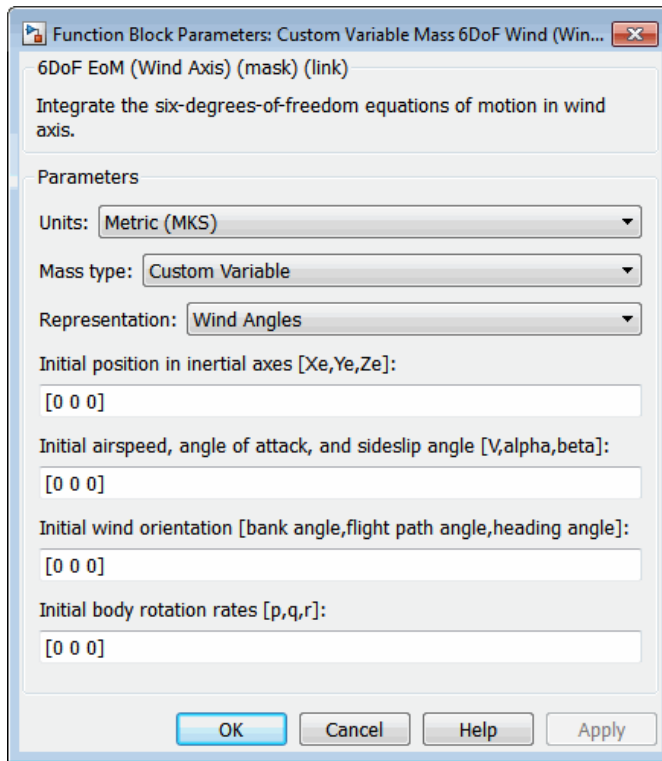
$$\begin{bmatrix} p_w \\ q_w \\ r_w \end{bmatrix} = DMC_{wb} \begin{bmatrix} p_b - \dot{\beta} \sin \alpha \\ q_b - \dot{\alpha} \\ r_b + \dot{\beta} \cos \alpha \end{bmatrix}$$

Using this relationship in the wind rate vector equations, gives the relationship between the wind rate vector and the body-fixed angular rates.

# Custom Variable Mass 6DoF Wind (Wind Angles)

$$\begin{bmatrix} \dot{\mu} \\ \dot{\gamma} \\ \dot{\chi} \end{bmatrix} = J \begin{bmatrix} p_w \\ q_w \\ r_w \end{bmatrix} = \begin{bmatrix} 1 & (\sin \mu \tan \gamma) & (\cos \mu \tan \gamma) \\ 0 & \cos \mu & -\sin \mu \\ 0 & \frac{\sin \mu}{\cos \gamma} & \frac{\cos \mu}{\cos \gamma} \end{bmatrix} DMC_{wb} \begin{bmatrix} p_b - \dot{\beta} \sin \alpha \\ q_b - \dot{\alpha} \\ r_b + \dot{\beta} \cos \alpha \end{bmatrix}$$

## Dialog Box



## Units

Specifies the input and output units:

# Custom Variable Mass 6DoF Wind (Wind Angles)

<b>Units</b>	<b>Forces</b>	<b>Moment</b>	<b>Acceleration</b>	<b>Velocity</b>	<b>Position</b>	<b>Mass</b>	<b>Inertia</b>
Metric (MKS)	Newton	Newton meter	Meters per second squared	Meters per second	Meters	Kilogram	Kilogram meter squared
English (Velocity in ft/s)	Pound	Foot pound	Feet per second squared	Feet per second	Feet	Slug	Slug foot squared
English (Velocity in kts)	Pound	Foot pound	Feet per second squared	Knots	Feet	Slug	Slug foot squared

## Mass Type

Select the type of mass to use:

Fixed	Mass is constant throughout the simulation.
Simple Variable	Mass and inertia vary linearly as a function of mass rate.
Custom Variable	Mass and inertia variations are customizable.

The Custom Variable selection conforms to the previously described equations of motion.

## Representation

Select the representation to use:

Wind Angles	Use wind angles within equations of motion.
Quaternion	Use quaternions within equations of motion.

# Custom Variable Mass 6DoF Wind (Wind Angles)

---

The Wind Angles selection conforms to the previously described equations of motion.

## **Initial position in inertial axes**

The three-element vector for the initial location of the body in the flat Earth reference frame.

## **Initial airspeed, sideslip angle, and angle of attack**

The three-element vector containing the initial airspeed, initial sideslip angle and initial angle of attack.

## **Initial wind orientation**

The three-element vector containing the initial wind angles [bank, flight path, and heading], in radians.

## **Initial body rotation rates**

The three-element vector for the initial body-fixed angular rates, in radians per second.

## **Inputs and Outputs**

<b>Input</b>	<b>Dimension Type</b>	<b>Description</b>
First	Vector	Contains the three applied forces in wind-fixed axes.
Second	Vector	Contains the three applied moments in body-fixed axes.
Third	Scalar	Contains the rate of change of mass.
Fourth	Scalar	Contains the mass.
Fifth	3-by-3 matrix	Applies to the rate of change of inertia tensor matrix in body-fixed axes.
Sixth	3-by-3 matrix	Applies to the inertia tensor matrix in body-fixed axes.

# Custom Variable Mass 6DoF Wind (Wind Angles)

Output	Dimension Type	Description
First	Three-element vector	Contains the velocity in the flat Earth reference frame.
Second	Three-element vector	Contains the position in the flat Earth reference frame.
Third	Three-element vector	Contains the wind rotation angles [bank, flight path, heading], in radians.
Fourth	3-by-3 matrix	Applies to the coordinate transformation from flat Earth axes to wind-fixed axes.
Fifth	Three-element vector	Contains the velocity in the wind-fixed frame.
Sixth	Two-element vector	Contains the angle of attack and sideslip angle, in radians.
Seventh	Two-element vector	Contains the rate of change of angle of attack and rate of change of sideslip angle, in radians per second.
Eighth	Three-element vector	Contains the angular rates in body-fixed axes, in radians per second.
Ninth	Three-element vector	Contains the angular accelerations in body-fixed axes, in radians per second squared.
Tenth	Three-element vector	Contains the accelerations in body-fixed axes.

## Assumptions and Limitations

The block assumes that the applied forces are acting at the center of gravity of the body.

# Custom Variable Mass 6DoF Wind (Wind Angles)

---

## References

Mangiacasale, L., *Flight Mechanics of a  $\mu$ -Airplane with a MATLAB Simulink Helper*, Edizioni Libreria CLUP, Milan, 1998.

Stevens, B. L., and F. L. Lewis, *Aircraft Control and Simulation*, John Wiley & Sons, New York, 1992.

## See Also

6DoF (Euler Angles)

6DoF (Quaternion)

6DoF ECEF (Quaternion)

6DoF Wind (Quaternion)

6DoF Wind (Wind Angles)

6th Order Point Mass (Coordinated Flight)

Custom Variable Mass 6DoF (Euler Angles)

Custom Variable Mass 6DoF (Quaternion)

Custom Variable Mass 6DoF ECEF (Quaternion)

Custom Variable Mass 6DoF Wind (Quaternion)

Simple Variable Mass 6DoF (Euler Angles)

Simple Variable Mass 6DoF (Quaternion)

Simple Variable Mass 6DoF ECEF (Quaternion)

Simple Variable Mass 6DoF Wind (Quaternion)

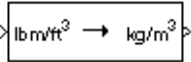
Simple Variable Mass 6DoF Wind (Wind Angles)



**Purpose** Convert from density units to desired density units

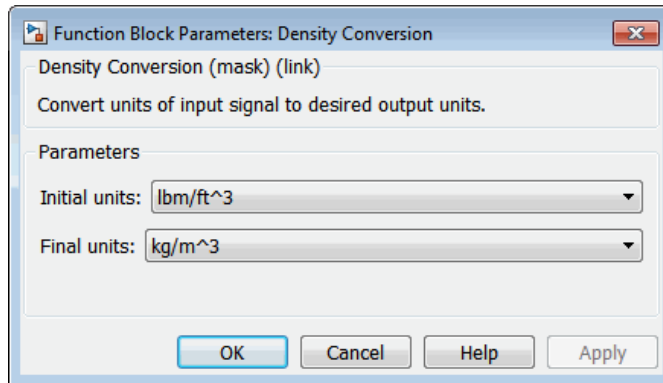
**Library** Utilities/Unit Conversions

**Description** The Density Conversion block computes the conversion factor from specified input density units to specified output density units and applies the conversion factor to the input signal.



The Density Conversion block icon displays the input and output units selected from the **Initial units** and the **Final units** lists.

## Dialog Box



**Initial units**  
Specifies the input units.

**Final units**  
Specifies the output units.

The following conversion units are available:

lbm/ft <sup>3</sup>	Pound mass per cubic foot
kg/m <sup>3</sup>	Kilograms per cubic meter

# Density Conversion

---

slug/ft<sup>3</sup>

Slugs per cubic foot

lbm/in<sup>3</sup>

Pound mass per cubic inch

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the density, in initial density units.

Output	Dimension Type	Description
First		Contains the density, in final density units.

## See Also

Acceleration Conversion

Angle Conversion

Angular Acceleration Conversion

Angular Velocity Conversion

Force Conversion

Length Conversion

Mass Conversion

Pressure Conversion

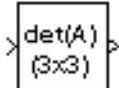
Temperature Conversion

Velocity Conversion

**Purpose** Compute determinant of matrix

**Library** Utilities/Math Operations

**Description** The Determinant of 3x3 Matrix block computes the determinant for the input matrix.



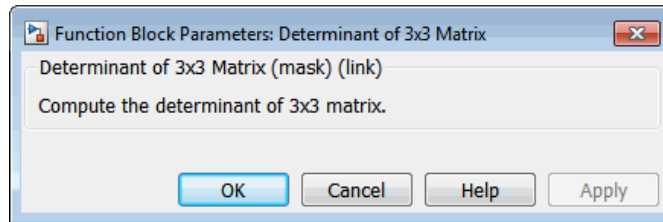
The input matrix has the form of

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}$$

The determinant of the matrix has the form of

$$\det(A) = A_{11}(A_{22}A_{33} - A_{23}A_{32}) - A_{12}(A_{21}A_{33} - A_{23}A_{31}) + A_{13}(A_{21}A_{32} - A_{22}A_{31})$$

## Dialog Box



## Inputs and Outputs

Input	Dimension Type	Description
First	3-by-3 matrix	

Output	Dimension Type	Description
First		Contains the determinant of input matrix.

**See Also** Adjoint of 3x3 Matrix

# Determinant of 3x3 Matrix

---

Create 3x3 Matrix

Invert 3x3 Matrix

# Digital DATCOM Forces and Moments

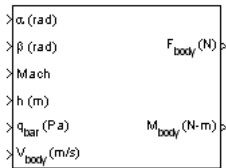
## Purpose

Compute aerodynamic forces and moments using Digital DATCOM static and dynamic stability derivatives

## Library

Aerodynamics

## Description



The Digital DATCOM Forces and Moments block computes the aerodynamic forces and moments about the center of gravity using aerodynamic coefficients from Digital DATCOM.

Algorithms for calculating forces and moments build up the overall aerodynamic forces and moments ( $F$  and  $M$ ) from data contained in the **Digital DATCOM structure** parameter:

$$F = F_{\text{static}} + F_{\text{dyn}}$$

$$M = M_{\text{static}} + M_{\text{dyn}}$$

$F_{\text{static}}$  and  $M_{\text{static}}$  are the static contribution, and  $F_{\text{dyn}}$  and  $M_{\text{dyn}}$  the dynamic contribution, to the aerodynamic coefficients. If the dynamic characteristics are not contained in the **Digital DATCOM structure** parameter, their contribution is set to zero.

## Static Stability Characteristics

Static stability characteristics include the following.

### Coefficient Meaning

$C_D$	Matrix of drag coefficients. These coefficients are defined positive for an aft-acting load.
$C_L$	Matrix of lift coefficients. These coefficients are defined positive for an up-acting load.
$C_m$	Matrix of pitching-moment coefficients. These coefficients are defined positive for a nose-up rotation.
$C_{Y\beta}$	Matrix of derivatives of side-force coefficients with respect to sideslip angle

# Digital DATCOM Forces and Moments

---

## **Coefficient    Meaning**

$C_{n\beta}$	Matrix of derivatives of yawing-moment coefficients with respect to sideslip angle
$C_{l\beta}$	Matrix of derivatives of rolling-moment coefficients with respect to sideslip angle

These are the static contributions to the aerodynamic coefficients in stability axes.

$$C_{D \text{ static}} = C_D$$

$$C_{y \text{ static}} = C_{Y\beta}\beta$$

$$C_{L \text{ static}} = C_L$$

$$C_{l \text{ static}} = C_{l\beta}\beta$$

$$C_{m \text{ static}} = C_M$$

$$C_{n \text{ static}} = C_{n\beta}\beta$$

## **Dynamic Stability Characteristics**

Dynamic stability characteristics include the following.

### **Coefficient    Meaning**

$C_{lq}$	Matrix of rolling-moment derivatives due to pitch rate
$C_{mq}$	Matrix of pitching-moment derivatives due to pitch rate
$C_{Lda/dt}$	Matrix of lift force derivatives due to rate of angle of attack
$C_{mda/dt}$	Matrix of pitching-moment derivatives due to rate of angle of attack
$C_{lp}$	Matrix of rolling-moment derivatives due to roll rate

## Coefficient    Meaning

$C_{Yp}$             Matrix of lateral force derivatives due to roll rate

$C_{np}$             Matrix of yawing-moment derivatives due to roll rate

$C_{nr}$             Matrix of yawing-moment derivatives due to yaw rate

$C_{lr}$             Matrix of rolling-moment derivatives due to yaw rate

These are the dynamic contributions to the aerodynamic coefficients in stability axes.

$$C_{D \text{ dyn}} = 0$$

$$C_{Y \text{ dyn}} = C_{yp} p (b_{\text{ref}} / 2V)$$

$$C_{L \text{ dyn}} = C_{L\dot{a}} \dot{a} (c_{\text{bar}} / 2V)$$

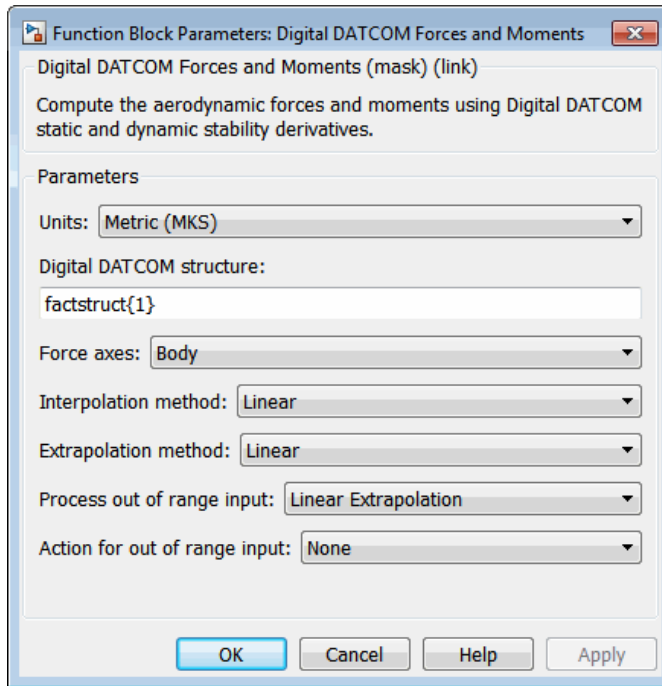
$$C_{l \text{ dyn}} = (C_{lp} p + C_{lq} q + C_{lr} r) (b_{\text{ref}} / 2V)$$

$$C_{m \text{ dyn}} = (C_{mq} q + C_{m\dot{a}} \dot{a}) (c_{\text{bar}} / 2V)$$

$$C_{n \text{ dyn}} = (C_{np} p + C_{nr} r) (b_{\text{ref}} / 2V)$$

# Digital DATCOM Forces and Moments

## Dialog Box



### Units

Specifies the input and output units:

Units	Force	Moment	Length	Velocity	Pressure
Metric (MKS)	Newton	Newton-meter	Meters	Meters per second	Pascal
English (Velocity in ft/s)	Pound	Foot-pound	Feet	Feet per second	Pound per square inch
English (Velocity in kts)	Pound	Foot-pound	Feet	Knots	Pound per square inch



# Digital DATCOM Forces and Moments

## Digital DATCOM structure

Specifies the MATLAB structure containing the digital DATCOM data. This structure is generated by the Aerospace Toolbox function `datcomimport`. The Digital DATCOM Forces and Moments block supports only Digital DATCOM, which is the 1976 version of DATCOM.

## Force axes

Specifies coordinate system for aerodynamic force: `Body` or `Wind`.

## Interpolation method

`None (flat)` or `Linear`

## Extrapolation method

`None (clip)` or `Linear`

## Process out of range input

Specifies how to handle out-of-range input: `Linear Extrapolation` or `Clip to Range`.

## Action for out of range input

Specifies if out-of-range input invokes a warning, an error, or no action.

## Inputs and Outputs

Input	Dimension Type	Description
First	3-by-3 matrix	Contains the angle of attack.
Second		Contains the sideslip angle, in radians.
Third		Contains the Mach number.
Fourth		Contains the altitude, in selected length units.
Fifth		Contains the dynamic pressure, in selected pressure units.
Sixth		Contains the velocity, selected velocity units and selected force axes.

# Digital DATCOM Forces and Moments

Input	Dimension Type	Description
Seventh (Optional)		Contains the angle of attack rate, in radians per second. Appears when DAMP Control Card is used in input to Digital DATCOM.
Eight (Optional)		Contains the body angular rates, in radians per second. Appears when DAMP Control Card is used in input to Digital DATCOM.
Ninth (Optional)		Contains the ground height, in select units of length. Appears when GRNDEF Namelist is used in input to Digital DATCOM.
Tenth (Optional)		Contains the control surface deflections, radians. Appears when ASYFLP or SYMFLP and GRNDEF namelists are used in input to Digital DATCOM.

Output	Dimension Type	Description
First		Contains the aerodynamic forces at the center of gravity in selected coordinate system: Body ( $F_x$ , $F_y$ , and $F_z$ ), or Wind ( $F_D$ , $F_y$ , and $F_L$ ).
Second		Contains the aerodynamic moments at the center of gravity in body coordinates ( $M_x$ , $M_y$ , and $M_z$ ).

## Assumptions and Limitations

The operational limitations of Digital DATCOM apply to the data contained in the **Digital DATCOM structure** parameter. For more information on Digital DATCOM limitations, see Section 2.4.5 of reference [1].

# Digital DATCOM Forces and Moments

---

The **Digital DATCOM structure** parameters `alpha`, `mach`, `alt`, `grndht`, and `delta` must be strictly monotonically increasing to be used with the Digital DATCOM Forces and Moments block.

The **Digital DATCOM structure** coefficients must correspond to the dimensions of the breakpoints (`alpha`, `mach`, `alt`, `grndht`, and `delta`) to be used with the Digital DATCOM Forces and Moments block.

## References

[1] *The USAF Stability and Control Digital Datcom*, AFFDL-TR-79-3032, 1979.

[2] Etkin, B., and L. D. Reid, *Dynamics of Flight Stability and Control*, John Wiley & Sons, New York, 1996.

[3] Roskam, J., "Airplane Design Part VI: Preliminary Calculation of Aerodynamic, Thrust and Power Characteristics," Roskam Aviation and Engineering Corporation, Ottawa, Kansas, 1987.

[4] Stevens, B. L., and F. L. Lewis, *Aircraft Control and Simulation*, John Wiley & Sons, New York, 1992.

## See Also

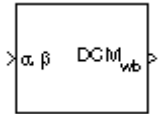
Aerodynamic Forces and Moments

# Direction Cosine Matrix Body to Wind

**Purpose** Convert angle of attack and sideslip angle to direction cosine matrix

**Library** Utilities/Axes Transformations

## Description



The Direction Cosine Matrix Body to Wind block converts angle of attack and sideslip angle into a 3-by-3 direction cosine matrix (DCM). The DCM matrix performs the coordinate transformation of a vector in body axes ( $ox_0, oy_0, oz_0$ ) into a vector in wind axes ( $ox_2, oy_2, oz_2$ ). The order of the axis rotations required to bring this about is:

- 1 A rotation about  $oy_0$  through the angle of attack ( $\alpha$ ) to axes ( $ox_1, oy_1, oz_1$ )
- 2 A rotation about  $oz_1$  through the sideslip angle ( $\beta$ ) to axes ( $ox_2, oy_2, oz_2$ )

$$\begin{bmatrix} ox_2 \\ oy_2 \\ oz_2 \end{bmatrix} = DCM_{wb} \begin{bmatrix} ox_0 \\ oy_0 \\ oz_0 \end{bmatrix}$$

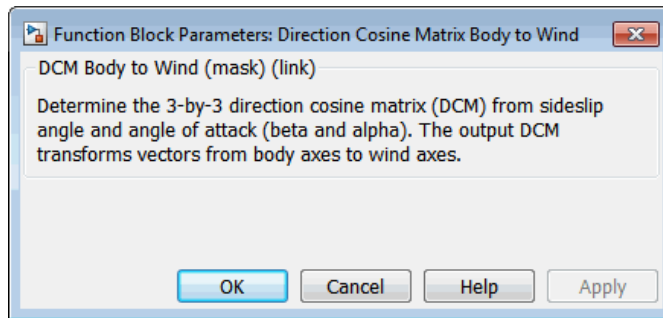
$$\begin{bmatrix} ox_2 \\ oy_2 \\ oz_2 \end{bmatrix} = \begin{bmatrix} \cos \beta & \sin \beta & 0 \\ -\sin \beta & \cos \beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \alpha & 0 & \sin \alpha \\ 0 & 1 & 0 \\ -\sin \alpha & 0 & \cos \alpha \end{bmatrix} \begin{bmatrix} ox_0 \\ oy_0 \\ oz_0 \end{bmatrix}$$

Combining the two axis transformation matrices defines the following DCM.

$$DCM_{wb} = \begin{bmatrix} \cos \alpha \cos \beta & \sin \beta & \sin \alpha \cos \beta \\ -\cos \alpha \sin \beta & \cos \beta & -\sin \alpha \sin \beta \\ -\sin \alpha & 0 & \cos \alpha \end{bmatrix}$$

# Direction Cosine Matrix Body to Wind

## Dialog Box



## Inputs and Outputs

Input	Dimension Type	Description
First	2-by-1 vector	Contains the angle of attack and sideslip angle, in radians.

Output	Dimension Type	Description
First	3-by-3 direction cosine matrix	Transforms body-fixed vectors to wind-fixed vectors.

## Reference

Stevens, B. L., and F. L. Lewis, *Aircraft Control and Simulation*, John Wiley & Sons, New York, 1992.

## See Also

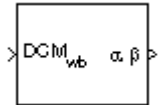
Direction Cosine Matrix Body to Wind to Alpha and Beta  
Direction Cosine Matrix to Rotation Angles  
Direction Cosine Matrix to Wind Angles  
Rotation Angles to Direction Cosine Matrix  
Wind Angles to Direction Cosine Matrix

# Direction Cosine Matrix Body to Wind to Alpha and Beta

**Purpose** Convert direction cosine matrix to angle of attack and sideslip angle

**Library** Utilities/Axes Transformations

## Description



The Direction Cosine Matrix Body to Wind to Alpha and Beta block converts a 3-by-3 direction cosine matrix (DCM) into angle of attack and sideslip angle. The DCM matrix performs the coordinate transformation of a vector in body axes ( $ox_0, oy_0, oz_0$ ) into a vector in wind axes ( $ox_2, oy_2, oz_2$ ). The order of the axis rotations required to bring this about is:

- 1 A rotation about  $oy_0$  through the angle of attack ( $\alpha$ ) to axes ( $ox_1, oy_1, oz_1$ )
- 2 A rotation about  $oz_1$  through the sideslip angle ( $\beta$ ) to axes ( $ox_2, oy_2, oz_2$ )

$$\begin{bmatrix} ox_2 \\ oy_2 \\ oz_2 \end{bmatrix} = DCM_{wb} \begin{bmatrix} ox_0 \\ oy_0 \\ oz_0 \end{bmatrix}$$

$$\begin{bmatrix} ox_2 \\ oy_2 \\ oz_2 \end{bmatrix} = \begin{bmatrix} \cos \beta & \sin \beta & 0 \\ -\sin \beta & \cos \beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \alpha & 0 & \sin \alpha \\ 0 & 1 & 0 \\ -\sin \alpha & 0 & \cos \alpha \end{bmatrix} \begin{bmatrix} ox_0 \\ oy_0 \\ oz_0 \end{bmatrix}$$

Combining the two axis transformation matrices defines the following DCM.

$$DCM_{wb} = \begin{bmatrix} \cos \alpha \cos \beta & \sin \beta & \sin \alpha \cos \beta \\ -\cos \alpha \sin \beta & \cos \beta & -\sin \alpha \sin \beta \\ -\sin \alpha & 0 & \cos \alpha \end{bmatrix}$$

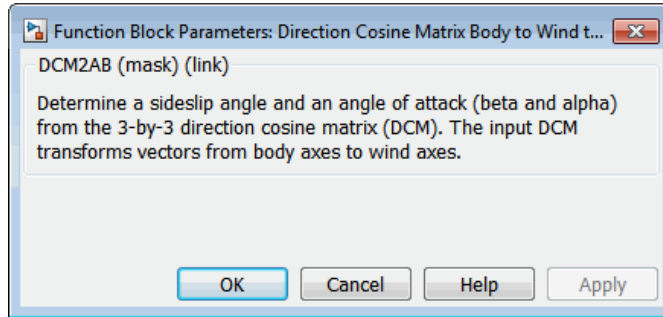
To determine angles from the DCM, the following equations are used:

# Direction Cosine Matrix Body to Wind to Alpha and Beta

$$\alpha = \text{asin}(-DCM(3,1))$$

$$\beta = \text{asin}(DCM(1,2))$$

## Dialog Box



## Inputs and Outputs

Input	Dimension Type	Description
First	3-by-3 direction cosine matrix	Transforms body-fixed vectors to wind-fixed vectors.

Output	Dimension Type	Description
First	2-by-1 vector	Contains angle of attack and sideslip angle, in radians.

## Assumptions and Limitations

This implementation generates angles that lie between  $\pm 90$  degrees.

## Reference

Stevens, B. L., and F. L. Lewis, *Aircraft Control and Simulation*, John Wiley & Sons, New York, 1992.

## See Also

Direction Cosine Matrix Body to Wind  
Direction Cosine Matrix to Rotation Angles

# Direction Cosine Matrix Body to Wind to Alpha and Beta

---

Direction Cosine Matrix to Wind Angles

Rotation Angles to Direction Cosine Matrix

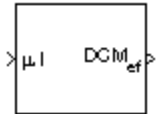
Wind Angles to Direction Cosine Matrix



**Purpose** Convert geodetic latitude and longitude to direction cosine matrix

**Library** Utilities/Axes Transformations

**Description**



The Direction Cosine Matrix ECEF to NED block converts geodetic latitude and longitude into a 3-by-3 direction cosine matrix (DCM). The DCM matrix performs the coordinate transformation of a vector in Earth-centered Earth-fixed (ECEF) axes ( $ox_0, oy_0, oz_0$ ) into a vector in north-east-down (NED) axes ( $ox_2, oy_2, oz_2$ ). The order of the axis rotations required to bring this about is:

- 1 A rotation about  $oz_0$  through the longitude ( $l$ ) to axes ( $ox_1, oy_1, oz_1$ )
- 2 A rotation about  $oy_1$  through the geodetic latitude ( $\mu$ ) to axes ( $ox_2, oy_2, oz_2$ )

$$\begin{bmatrix} ox_2 \\ oy_2 \\ oz_2 \end{bmatrix} = DCM_{ef} \begin{bmatrix} ox_0 \\ oy_0 \\ oz_0 \end{bmatrix}$$

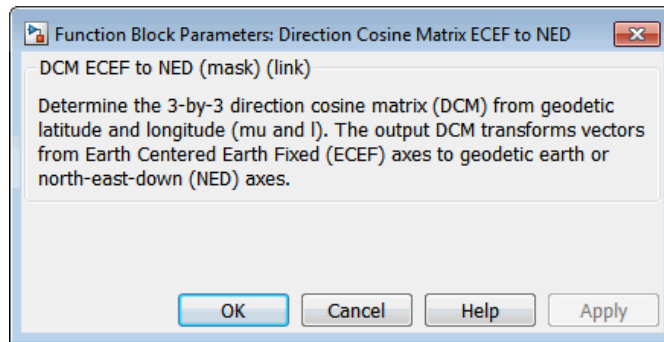
$$\begin{bmatrix} ox_2 \\ oy_2 \\ oz_2 \end{bmatrix} = \begin{bmatrix} -\sin \mu & 0 & \cos \mu \\ 0 & 1 & 0 \\ -\cos \mu & 0 & -\sin \mu \end{bmatrix} \begin{bmatrix} \cos l & \sin l & 0 \\ -\sin l & \cos l & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} ox_0 \\ oy_0 \\ oz_0 \end{bmatrix}$$

Combining the two axis transformation matrices defines the following DCM.

$$DCM_{ef} = \begin{bmatrix} -\sin \mu \cos l & -\sin \mu \sin l & \cos \mu \\ -\sin l & \cos l & 0 \\ -\cos \mu \cos l & -\cos \mu \sin l & -\sin \mu \end{bmatrix}$$

# Direction Cosine Matrix ECEF to NED

## Dialog Box



## Inputs and Outputs

Input	Dimension Type	Description
First	2-by-1 vector	Contains the geodetic latitude and longitude, in degrees.

Output	Dimension Type	Description
First	3-by-3 direction cosine matrix	Transforms ECEF vectors to NED vectors.

## Assumptions

The implementation of the ECEF coordinate system assumes that the origin is at the center of the planet, the  $x$ -axis intersects the Greenwich meridian and the equator, the  $z$ -axis is the mean spin axis of the planet, positive to the north, and the  $y$ -axis completes the right-hand system.

## References

Stevens, B. L., and F. L. Lewis, *Aircraft Control and Simulation*, John Wiley & Sons, New York, 1992.

Zipfel, P. H., *Modeling and Simulation of Aerospace Vehicle Dynamics*, AIAA Education Series, Reston, Virginia, 2000.

“Atmospheric and Space Flight Vehicle Coordinate Systems,” ANSI/AIAA R-004-1992.

## See Also

Direction Cosine Matrix ECEF to NED to Latitude and Longitude

Direction Cosine Matrix to Rotation Angles

Direction Cosine Matrix to Wind Angles

ECEF Position to LLA

Rotation Angles to Direction Cosine Matrix

LLA to ECEF Position

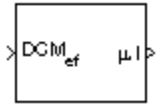
Wind Angles to Direction Cosine Matrix

# Direction Cosine Matrix ECEF to NED to Latitude and Longitude

**Purpose** Convert direction cosine matrix to geodetic latitude and longitude

**Library** Utilities/Axes Transformations

## Description



The Direction Cosine Matrix ECEF to NED to Latitude and Longitude block converts a 3-by-3 direction cosine matrix (DCM) into geodetic latitude and longitude. The DCM matrix performs the coordinate transformation of a vector in Earth-centered Earth-fixed (ECEF) axes ( $ox_0, oy_0, oz_0$ ) into a vector in north-east-down (NED) axes ( $ox_2, oy_2, oz_2$ ). The order of the axis rotations required to bring this about is:

- 1 A rotation about  $oz_0$  through the longitude ( $l$ ) to axes ( $ox_1, oy_1, oz_1$ )
- 2 A rotation about  $oy_1$  through the geodetic latitude ( $\mu$ ) to axes ( $ox_2, oy_2, oz_2$ )

$$\begin{bmatrix} ox_2 \\ oy_2 \\ oz_2 \end{bmatrix} = DCM_{ef} \begin{bmatrix} ox_0 \\ oy_0 \\ oz_0 \end{bmatrix}$$

$$\begin{bmatrix} ox_2 \\ oy_2 \\ oz_2 \end{bmatrix} = \begin{bmatrix} -\sin \mu & 0 & \cos \mu \\ 0 & 1 & 0 \\ -\cos \mu & 0 & -\sin \mu \end{bmatrix} \begin{bmatrix} \cos l & \sin l & 0 \\ -\sin l & \cos l & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} ox_0 \\ oy_0 \\ oz_0 \end{bmatrix}$$

Combining the two axis transformation matrices defines the following DCM.

$$DCM_{ef} = \begin{bmatrix} -\sin \mu \cos l & -\sin \mu \sin l & \cos \mu \\ -\sin l & \cos l & 0 \\ -\cos \mu \cos l & -\cos \mu \sin l & -\sin \mu \end{bmatrix}$$

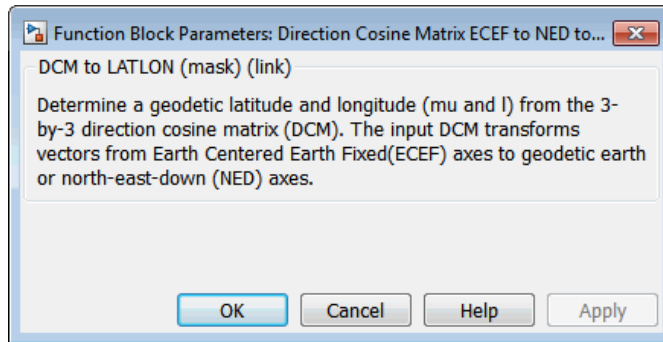
To determine geodetic latitude and longitude from the DCM, the following equations are used:

# Direction Cosine Matrix ECEF to NED to Latitude and Longitude

$$\mu = \text{asin}(-DCM(3,3))$$

$$l = \text{atan}\left(\frac{-DCM(2,1)}{DCM(2,2)}\right)$$

## Dialog Box



## Inputs and Outputs

Input	Dimension Type	Description
First	3-by-3 direction cosine matrix	Transforms ECEF vectors to NED vectors.

Output	Dimension Type	Description
First	2-by-1 vector	Contains the geodetic latitude and longitude, in degrees.

## Assumptions and Limitations

This implementation generates a geodetic latitude that lies between  $\pm 90$  degrees, and longitude that lies between  $\pm 180$  degrees.

The implementation of the ECEF coordinate system assumes that the origin is at the center of the planet, the  $x$ -axis intersects the Greenwich meridian and the equator, the  $z$ -axis is the mean spin axis of the planet, positive to the north, and the  $y$ -axis completes the right-hand system.

# Direction Cosine Matrix ECEF to NED to Latitude and Longitude

---

## References

Stevens, B. L., and F. L. Lewis, *Aircraft Control and Simulation*, John Wiley & Sons, New York, 1992.

Zipfel, P. H., *Modeling and Simulation of Aerospace Vehicle Dynamics*, AIAA Education Series, Reston, Virginia, 2000.

“Atmospheric and Space Flight Vehicle Coordinate Systems,” ANSI/AIAA R-004-1992.

## See Also

Direction Cosine Matrix ECEF to NED

Direction Cosine Matrix to Rotation Angles

Direction Cosine Matrix to Wind Angles

ECEF Position to LLA

Rotation Angles to Direction Cosine Matrix

LLA to ECEF Position

Wind Angles to Direction Cosine Matrix

# Direction Cosine Matrix to Quaternions

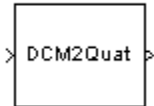
## Purpose

Convert direction cosine matrix to quaternion vector

## Library

Utilities/Axes Transformations

## Description



The Direction Cosine Matrix to Quaternions block transforms a 3-by-3 direction cosine matrix (DCM) into a four-element unit quaternion vector  $(q_0, q_1, q_2, q_3)$ . The DCM performs the coordinate transformation of a vector in inertial axes to a vector in body axes.

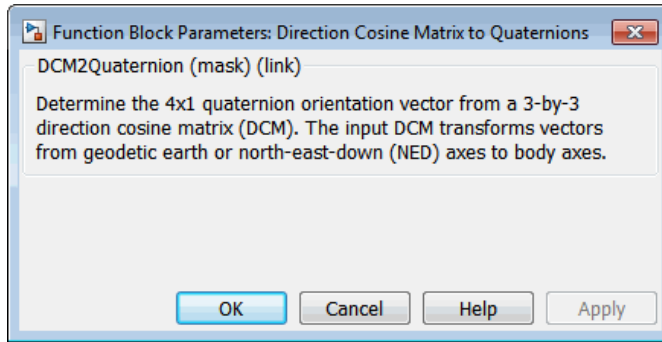
The DCM is defined as a function of a unit quaternion vector by the following:

$$DCM = \begin{bmatrix} (q_0^2 + q_1^2 - q_2^2 - q_3^2) & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & (q_0^2 - q_1^2 + q_2^2 - q_3^2) & 2(q_2q_3 + q_0q_1) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & (q_0^2 - q_1^2 - q_2^2 + q_3^2) \end{bmatrix}$$

Using this representation of the DCM, there are a number of calculations to arrive at the correct quaternion. The first of these is to calculate the trace of the DCM to determine which algorithms are used. If the trace is greater than zero, the quaternion can be automatically calculated. When the trace is less than or equal to zero, the major diagonal element of the DCM with the greatest value must be identified to determine the final algorithm used to calculate the quaternion. Once the major diagonal element is identified, the quaternion is calculated. For a detailed view of these algorithms, look under the mask of this block.

# Direction Cosine Matrix to Quaternions

## Dialog Box



## Inputs and Outputs

Input	Dimension Type	Description
First	3-by-3 direction cosine matrix	

Output	Dimension Type	Description
First	4-by-1 quaternion vector	

## See Also

Direction Cosine Matrix to Rotation Angles  
Rotation Angles to Direction Cosine Matrix  
Rotation Angles to Quaternions  
Quaternions to Direction Cosine Matrix  
Quaternions to Rotation Angles



# Direction Cosine Matrix to Rotation Angles

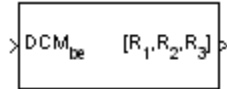
## Purpose

Convert direction cosine matrix to rotation angles

## Library

Utilities/Axes Transformations

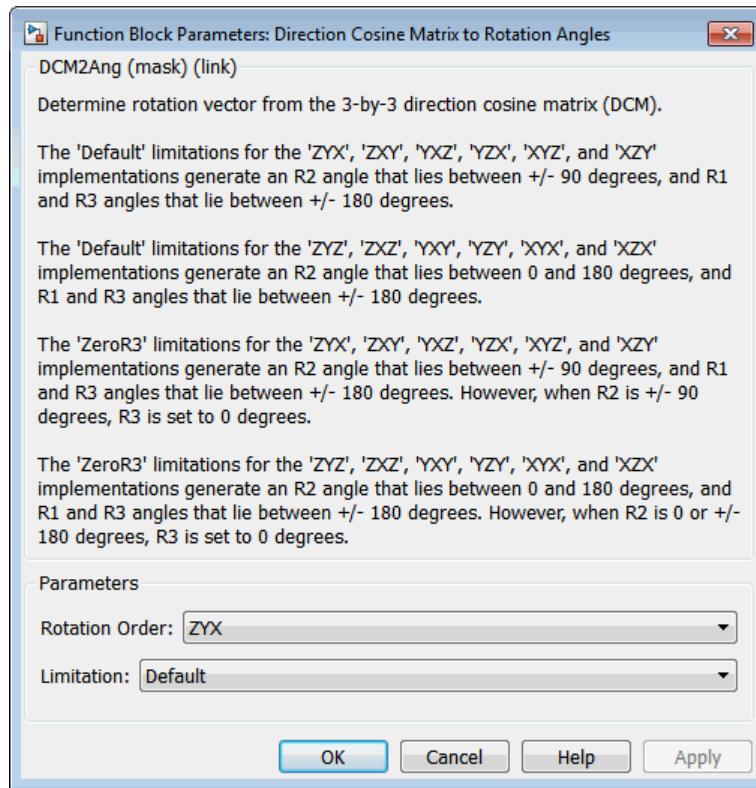
## Description



The Direction Cosine Matrix to Rotation Angles block converts a 3-by-3 direction cosine matrix (DCM) into three rotation angles R1, R2, and R3, respectively the first, second, and third rotation angles. The DCM matrix performs the coordinate transformation of a vector in inertial axes into a vector in body axes. The block **Rotation Order** parameter specifies the order of the block output rotations. For example, if **Rotation Order** has a value of ZYX, the block outputs are in the rotation order z-y-x (psi theta phi).

# Direction Cosine Matrix to Rotation Angles

## Dialog Box



### Rotation Order

Specifies the output rotation order for three rotation angles. From the list, select ZYX, ZYZ, ZXY, ZXZ, YXZ, YXY, YZX, YZY, XYZ, XYX, XZY, or XZX. The default is ZYX.

### Limitation

The 'Default' limitations for the 'ZYX', 'ZXY', 'YXZ', 'YZX', 'XYZ', and 'XZY' implementations generate an R2 angle that lies between  $\pm 90$  degrees, and R1 and R3 angles that lie between  $\pm 180$  degrees.

# Direction Cosine Matrix to Rotation Angles

The 'Default' limitations for the 'ZYZ', 'ZXZ', 'YXY', 'YZY', 'YXX', and 'XZX' implementations generate an R2 angle that lies between 0 and 180 degrees, and R1 and R3 angles that lie between  $\pm 180$  degrees.

The 'ZeroR3' limitations for the 'ZYX', 'ZXY', 'YXZ', 'YZX', 'XYZ', and 'XZY' implementations generate an R2 angle that lies between  $\pm 90$  degrees, and R1 and R3 angles that lie between  $\pm 180$  degrees. However, when R2 is  $\pm 90$  degrees, R3 is set to 0 degrees.

The 'ZeroR3' limitations for the 'ZYZ', 'ZXZ', 'YXY', 'YZY', 'YXX', and 'XZX' implementations generate an R2 angle that lies between 0 and 180 degrees, and R1 and R3 angles that lie between  $\pm 180$  degrees. However, when R2 is 0 or  $\pm 180$  degrees, R3 is set to 0 degrees.

## Inputs and Outputs

Input	Dimension Type	Description
First	3-by-3 matrix	Contains the direction cosine matrix.

Output	Dimension Type	Description
First	3-by-1 vector	Contains the rotation angles, in radians.

## See Also

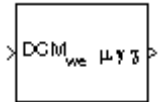
[Direction Cosine Matrix to Quaternions](#)  
[Quaternions to Direction Cosine Matrix](#)  
[Rotation Angles to Direction Cosine Matrix](#)

# Direction Cosine Matrix to Wind Angles

**Purpose** Convert direction cosine matrix to wind angles

**Library** Utilities/Axes Transformations

**Description**



The Direction Cosine Matrix to Wind Angles block converts a 3-by-3 direction cosine matrix (DCM) into three wind rotation angles. The DCM matrix performs the coordinate transformation of a vector in earth axes ( $ox_0, oy_0, oz_0$ ) into a vector in wind axes ( $ox_3, oy_3, oz_3$ ). The order of the axis rotations required to bring this about is:

- 1** A rotation about  $oz_0$  through the heading angle ( $\chi$ ) to axes ( $ox_1, oy_1, oz_1$ )
- 2** A rotation about  $oy_1$  through the flight path angle ( $\gamma$ ) to axes ( $ox_2, oy_2, oz_2$ )
- 3** A rotation about  $ox_2$  through the bank angle ( $\mu$ ) to axes ( $ox_3, oy_3, oz_3$ )

$$\begin{bmatrix} ox_3 \\ oy_3 \\ oz_3 \end{bmatrix} = DCM_{we} \begin{bmatrix} ox_0 \\ oy_0 \\ oz_0 \end{bmatrix}$$

$$\begin{bmatrix} ox_3 \\ oy_3 \\ oz_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \mu & \sin \mu \\ 0 & -\sin \mu & \cos \mu \end{bmatrix} \begin{bmatrix} \cos \gamma & 0 & -\sin \gamma \\ 0 & 1 & 0 \\ \sin \gamma & 0 & \cos \gamma \end{bmatrix} \begin{bmatrix} \cos \chi & \sin \chi & 0 \\ -\sin \chi & \cos \chi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} ox_0 \\ oy_0 \\ oz_0 \end{bmatrix}$$

Combining the three axis transformation matrices defines the following DCM.

$$DCM_{we} = \begin{bmatrix} \cos \gamma \cos \chi & \cos \gamma \sin \chi & -\sin \gamma \\ (\sin \mu \sin \gamma \cos \chi - \cos \mu \sin \chi) & (\sin \mu \sin \gamma \sin \chi + \cos \mu \cos \chi) & \sin \mu \cos \gamma \\ (\cos \mu \sin \gamma \cos \chi + \sin \mu \sin \chi) & (\cos \mu \sin \gamma \sin \chi - \sin \mu \cos \chi) & \cos \mu \cos \gamma \end{bmatrix}$$

To determine wind angles from the DCM, the following equations are used:

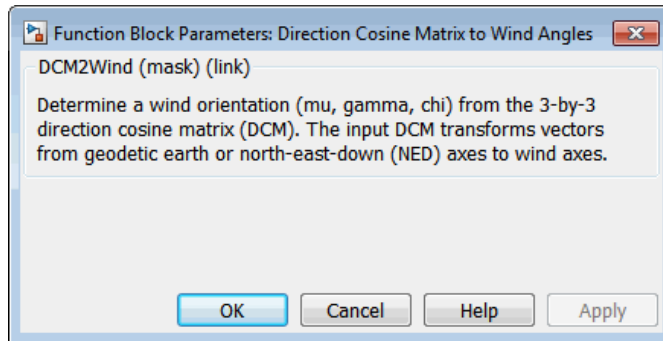
# Direction Cosine Matrix to Wind Angles

$$\mu = \operatorname{atan}\left(\frac{DCM(2,3)}{DCM(3,3)}\right)$$

$$\gamma = \operatorname{asin}(-DCM(1,3))$$

$$\chi = \operatorname{atan}\left(\frac{DCM(1,2)}{DCM(1,1)}\right)$$

## Dialog Box



## Inputs and Outputs

Input	Dimension Type	Description
First	3-by-3 direction cosine matrix	Transforms earth vectors to wind vectors.

Output	Dimension Type	Description
First	3-by-1 vector	Contains the wind angles, in radians.

## Assumptions and Limitations

This implementation generates a flight path angle that lies between  $\pm 90$  degrees, and bank and heading angles that lie between  $\pm 180$  degrees.

# Direction Cosine Matrix to Wind Angles

---

## **See Also**

Direction Cosine Matrix Body to Wind

Direction Cosine Matrix Body to Wind to Alpha and Beta

Direction Cosine Matrix to Rotation Angles

Rotation Angles to Direction Cosine Matrix

Wind Angles to Direction Cosine Matrix

**Purpose** Generate discrete wind gust

**Library** Environment/Wind

## Description

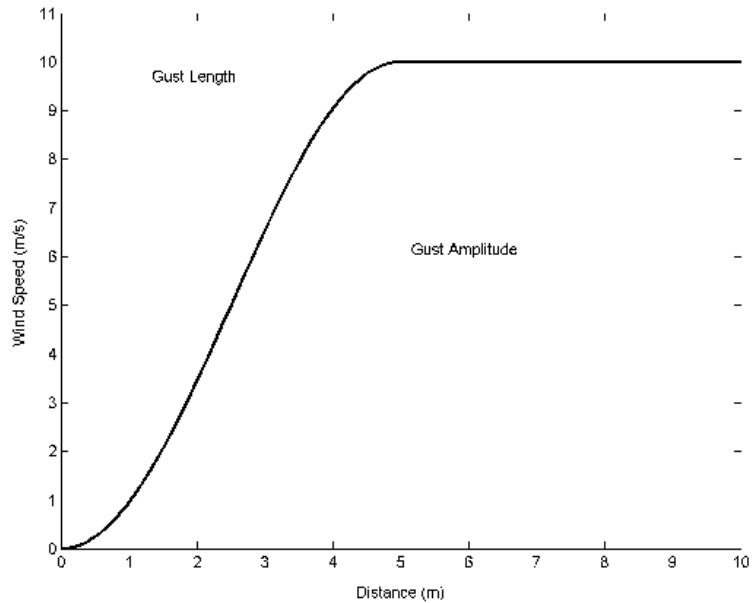


The Discrete Wind Gust Model block implements a wind gust of the standard “1-cosine” shape. This block implements the mathematical representation in the Military Specification MIL-F-8785C [1]. The gust is applied to each axis individually, or to all three axes at once. You specify the gust amplitude (the increase in wind speed generated by the gust), the gust length (length, in meters, over which the gust builds up) and the gust start time.

The Discrete Wind Gust Model block can represent the wind speed in units of feet per second, meters per second, or knots.

The following figure shows the shape of the gust with a start time of zero. The parameters that govern the gust shape are indicated on the diagram.

# Discrete Wind Gust Model



The discrete gust can be used singly or in multiples to assess airplane response to large wind disturbances.

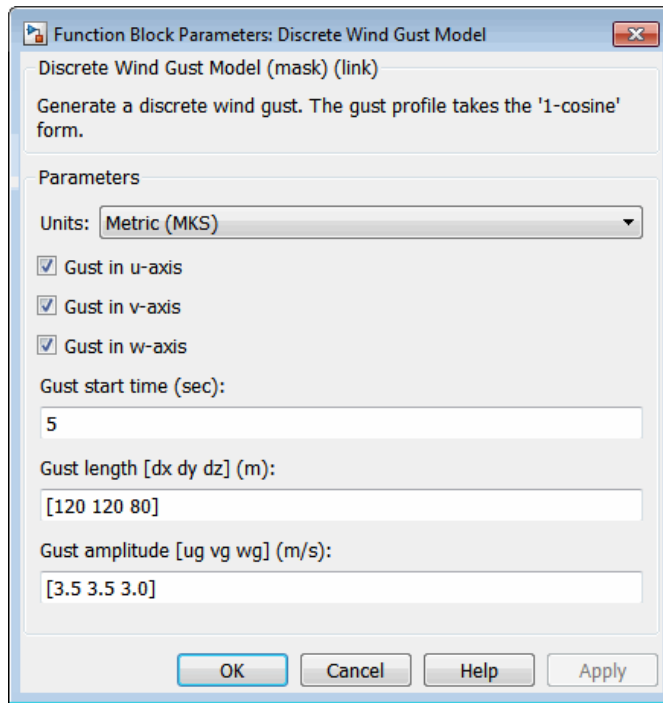
The mathematical representation of the discrete gust is

$$V_{wind} = \begin{cases} 0 & x < 0 \\ \frac{V_m}{2} \left( 1 - \cos \left( \frac{\pi x}{d_m} \right) \right) & 0 \leq x \leq d_m \\ V_m & x > d_m \end{cases}$$

where  $V_m$  is the gust amplitude,  $d_m$  is the gust length,  $x$  is the distance traveled, and  $V_{wind}$  is the resultant wind velocity in the body axis frame.



## Dialog Box



### Units

Define the units of wind gust.

Units	Wind	Altitude
Metric (MKS)	Meters/second	Meters
English (Velocity in ft/s)	Feet/second	Feet
English (Velocity in kts)	Knots	Feet

### Gust in u-axis

Select to apply the wind gust to the  $u$ -axis in the body frame.

# Discrete Wind Gust Model

---

## Gust in $v$ -axis

Select to apply the wind gust to the  $v$ -axis in the body frame.

## Gust in $w$ -axis

Select to apply the wind gust to the  $w$ -axis in the body frame.

## Gust start time (sec)

The model time, in seconds, at which the gust begins.

## Gust length [dx dy dz] (m or f)

The length, in meters or feet (depending on the choice of units), over which the gust builds up in each axis. These values must be positive.

## Gust amplitude [ug vg wg] (m/s, f/s, or knots)

The magnitude of the increase in wind speed caused by the gust in each axis. These values may be positive or negative.

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the airspeed in units selected.

Output	Dimension Type	Description
First		Contains the wind speed in units selected.

## Examples

See Discrete Wind Gust Model in aeroblk\_HL20 for an example of this block.

## Reference

U.S. Military Specification MIL-F-8785C, 5 November 1980.

## See Also

Dryden Wind Turbulence Model (Continuous)

Dryden Wind Turbulence Model (Discrete)

Von Karman Wind Turbulence Model (Continuous)

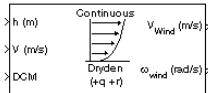
Wind Shear Model

# Dryden Wind Turbulence Model (Continuous)

**Purpose** Generate continuous wind turbulence with Dryden velocity spectra

**Library** Environment/Wind

**Description** The Dryden Wind Turbulence Model (Continuous) block uses the Dryden spectral representation to add turbulence to the aerospace model by passing band-limited white noise through appropriate forming filters. This block implements the mathematical representation in the Military Specification MIL-F-8785C and Military Handbook MIL-HDBK-1797.



According to the military references, turbulence is a stochastic process defined by velocity spectra. For an aircraft flying at a speed  $V$  through a frozen turbulence field with a spatial frequency of  $\Omega$  radians per meter, the circular frequency  $\omega$  is calculated by multiplying  $V$  by  $\Omega$ . The following table displays the component spectra functions:

	MIL-F-8785C	MIL-HDBK-1797
<b>Longitudinal</b>		
$\Phi_u(\omega)$	$\frac{2\sigma_u^2 L_u}{\pi V} \cdot \frac{1}{1 + \left(L_u \frac{\omega}{V}\right)^2}$	$\frac{2\sigma_u^2 L_u}{\pi V} \cdot \frac{1}{1 + \left(L_u \frac{\omega}{V}\right)^2}$
$\Phi_{p_g}(\omega)$	$\frac{\sigma_w^2}{VL_w} \cdot \frac{0.8 \left(\frac{\pi L_w}{4b}\right)^{1/3}}{1 + \left(\frac{4b\omega}{\pi V}\right)^2}$	$\frac{\sigma_w^2}{2VL_w} \cdot \frac{0.8 \left(\frac{2\pi L_w}{4b}\right)^{1/3}}{1 + \left(\frac{4b\omega}{\pi V}\right)^2}$
<b>Lateral</b>		

# Dryden Wind Turbulence Model (Continuous)

	<b>MIL-F-8785C</b>	<b>MIL-HDBK-1797</b>
$\Phi_v(\omega)$	$\frac{\sigma_v^2 L_v}{\pi V} \cdot \frac{1 + 3 \left( L_v \frac{\omega}{V} \right)^2}{\left[ 1 + \left( L_v \frac{\omega}{V} \right)^2 \right]^2}$	$\frac{2\sigma_v^2 L_v}{\pi V} \cdot \frac{1 + 12 \left( L_v \frac{\omega}{V} \right)^2}{\left[ 1 + 4 \left( L_v \frac{\omega}{V} \right)^2 \right]^2}$
$\Phi_r(\omega)$	$\frac{\mp \left( \frac{\omega}{V} \right)^2}{1 + \left( \frac{3b\omega}{\pi V} \right)^2} \cdot \Phi_v(\omega)$	$\frac{\mp \left( \frac{\omega}{V} \right)^2}{1 + \left( \frac{3b\omega}{\pi V} \right)^2} \cdot \Phi_v(\omega)$
<b>Vertical</b>		
$\Phi_w(\omega)$	$\frac{\sigma_w^2 L_w}{\pi V} \cdot \frac{1 + 3 \left( L_w \frac{\omega}{V} \right)^2}{\left[ 1 + \left( L_w \frac{\omega}{V} \right)^2 \right]^2}$	$\frac{2\sigma_w^2 L_w}{\pi V} \cdot \frac{1 + 12 \left( L_w \frac{\omega}{V} \right)^2}{\left[ 1 + 4 \left( L_w \frac{\omega}{V} \right)^2 \right]^2}$
$\Phi_q(\omega)$	$\frac{\pm \left( \frac{\omega}{V} \right)^2}{1 + \left( \frac{4b\omega}{\pi V} \right)^2} \cdot \Phi_w(\omega)$	$\frac{\pm \left( \frac{\omega}{V} \right)^2}{1 + \left( \frac{4b\omega}{\pi V} \right)^2} \cdot \Phi_w(\omega)$

The variable  $b$  represents the aircraft wingspan. The variables  $L_u$ ,  $L_v$ ,  $L_w$  represent the turbulence scale lengths. The variables  $\sigma_u$ ,  $\sigma_v$ ,  $\sigma_w$  represent the turbulence intensities.

# Dryden Wind Turbulence Model (Continuous)

---

The spectral density definitions of turbulence angular rates are defined in the specifications as three variations, which are displayed in the following table:

$$\begin{array}{lll} p_g = \frac{\partial w_g}{\partial y} & q_g = \frac{\partial w_g}{\partial x} & r_g = -\frac{\partial v_g}{\partial x} \\ p_g = \frac{\partial w_g}{\partial y} & q_g = \frac{\partial w_g}{\partial x} & r_g = \frac{\partial v_g}{\partial x} \\ p_g = -\frac{\partial w_g}{\partial y} & q_g = -\frac{\partial w_g}{\partial x} & r_g = \frac{\partial v_g}{\partial x} \end{array}$$

The variations affect only the vertical ( $q_g$ ) and lateral ( $r_g$ ) turbulence angular rates.

Keep in mind that the longitudinal turbulence angular rate spectrum

$$\Phi_{p_g}(\omega)$$

is a rational function. The rational function is derived from curve-fitting a complex algebraic function, not the vertical turbulence velocity spectrum,  $\Phi_w(\omega)$ , multiplied by a scale factor. Because the turbulence angular rate spectra contribute less to the aircraft gust response than the turbulence velocity spectra, it may explain the variations in their definitions.

The variations lead to the following combinations of vertical and lateral turbulence angular rate spectra:

# Dryden Wind Turbulence Model (Continuous)

## Vertical

$$\Phi_q(\omega)$$

$$\Phi_q(\omega)$$

$$-\Phi_q(\omega)$$

## Lateral

$$-\Phi_r(\omega)$$

$$\Phi_r(\omega)$$

$$\Phi_r(\omega)$$

To generate a signal with the correct characteristics, a unit variance, band-limited white noise signal is passed through forming filters. The forming filters are derived from the spectral square roots of the spectrum equations.

The following table displays the transfer functions:

	MIL-F-8785C	MIL-HDBK-1797
<b>Longitudinal</b>		
$H_u(s)$	$\sigma_u \sqrt{\frac{2L_u}{\pi V}} \cdot \frac{1}{1 + \frac{L_u}{V}s}$	$\sigma_u \sqrt{\frac{2L_u}{\pi V}} \cdot \frac{1}{1 + \frac{L_u}{V}s}$
$H_p(s)$	$\sigma_w \sqrt{\frac{0.8}{V}} \cdot \frac{\left(\frac{\pi}{4b}\right)^{1/6}}{L_w^{1/3} \left(1 + \left(\frac{4b}{\pi V}\right)s\right)}$	$\sigma_w \sqrt{\frac{0.8}{V}} \cdot \frac{\left(\frac{\pi}{4b}\right)^{1/6}}{(2L_w)^{1/3} \left(1 + \left(\frac{4b}{\pi V}\right)s\right)}$
<b>Lateral</b>		

# Dryden Wind Turbulence Model (Continuous)

	MIL-F-8785C	MIL-HDBK-1797
$H_v(s)$	$\sigma_v \sqrt{\frac{L_v}{\pi V}} \cdot \frac{1 + \frac{\sqrt{3}L_v s}{V}}{\left(1 + \frac{L_v s}{V}\right)^2}$	$\sigma_v \sqrt{\frac{2L_v}{\pi V}} \cdot \frac{1 + \frac{2\sqrt{3}L_v s}{V}}{\left(1 + \frac{2L_v s}{V}\right)^2}$
$H_r(s)$	$\frac{\mp \frac{s}{V}}{\left(1 + \left(\frac{3b}{\pi V}\right)s\right)} \cdot H_v(s)$	$\frac{\mp \frac{s}{V}}{\left(1 + \left(\frac{3b}{\pi V}\right)s\right)} \cdot H_v(s)$
<b>Vertical</b>		
$H_w(s)$	$\sigma_w \sqrt{\frac{L_w}{\pi V}} \cdot \frac{1 + \frac{\sqrt{3}L_w s}{V}}{\left(1 + \frac{L_w s}{V}\right)^2}$	$\sigma_w \sqrt{\frac{2L_w}{\pi V}} \cdot \frac{1 + \frac{2\sqrt{3}L_w s}{V}}{\left(1 + \frac{2L_w s}{V}\right)^2}$
$H_q(s)$	$\frac{\pm \frac{s}{V}}{\left(1 + \left(\frac{4b}{\pi V}\right)s\right)} \cdot H_w(s)$	$\frac{\pm \frac{s}{V}}{\left(1 + \left(\frac{4b}{\pi V}\right)s\right)} \cdot H_w(s)$

Divided into two distinct regions, the turbulence scale lengths and intensities are functions of altitude.



# Dryden Wind Turbulence Model (Continuous)

**Note** The military specifications result in the same transfer function after evaluating the turbulence scale lengths. The differences in turbulence scale lengths and turbulence transfer functions balance offset.

## Low-Altitude Model (Altitude < 1000 feet)

According to the military references, the turbulence scale lengths at low altitudes, where  $h$  is the altitude in feet, are represented in the following table:

MIL-F-8785C	MIL-HDBK-1797
$L_w = h$ $L_u = L_v = \frac{h}{(0.177 + 0.000823h)^{1.2}}$	$2L_w = h$ $L_u = 2L_v = \frac{h}{(0.177 + 0.000823h)^{1.2}}$

The turbulence intensities are given below, where  $W_{20}$  is the wind speed at 20 feet (6 m). Typically for light turbulence, the wind speed at 20 feet is 15 knots; for moderate turbulence, the wind speed is 30 knots; and for severe turbulence, the wind speed is 45 knots.

$$\sigma_w = 0.1W_{20}$$

$$\frac{\sigma_u}{\sigma_w} = \frac{\sigma_v}{\sigma_w} = \frac{1}{(0.177 + 0.000823h)^{0.4}}$$

The turbulence axes orientation in this region is defined as follows:

- Longitudinal turbulence velocity,  $u_g$ , aligned along the horizontal relative mean wind vector
- Vertical turbulence velocity,  $w_g$ , aligned with vertical

# Dryden Wind Turbulence Model (Continuous)

---

At this altitude range, the output of the block is transformed into body coordinates.

## Medium/High Altitudes (Altitude > 2000 feet)

For medium to high altitudes the turbulence scale lengths and intensities are based on the assumption that the turbulence is isotropic. In the military references, the scale lengths are represented by the following equations:

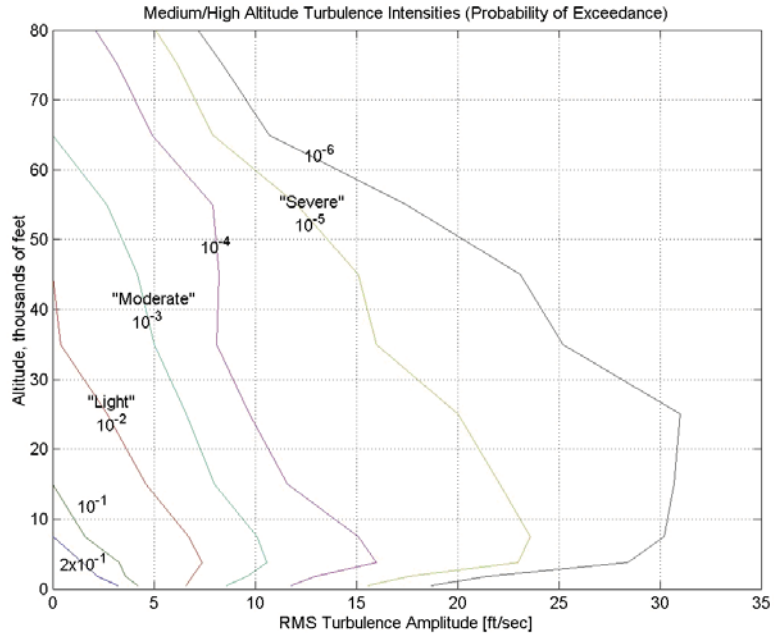
MIL-F-8785C	MIL-HDBK-1797
$L_u = L_v = L_w = 1750 \text{ ft}$	$L_u = 2L_v = 2L_w = 1750 \text{ ft}$

The turbulence intensities are determined from a lookup table that provides the turbulence intensity as a function of altitude and the probability of the turbulence intensity being exceeded. The relationship of the turbulence intensities is represented in the following equation:

$$\sigma_u = \sigma_v = \sigma_w.$$

The turbulence axes orientation in this region is defined as being aligned with the body coordinates.

# Dryden Wind Turbulence Model (Continuous)

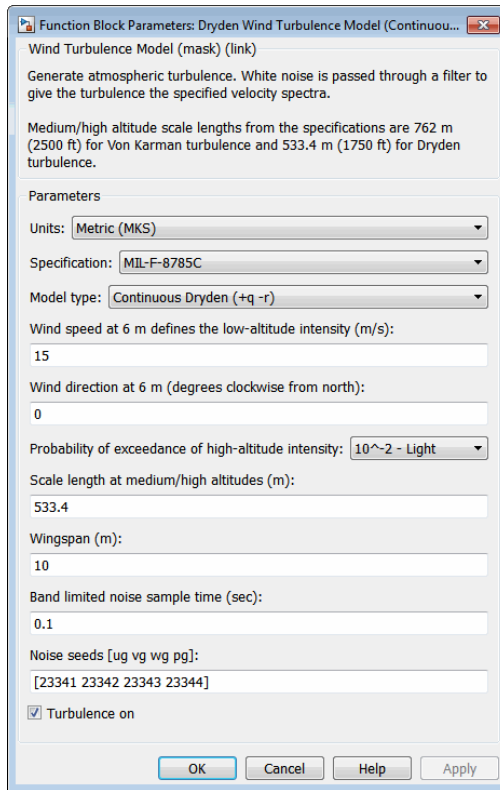


## Between Low and Medium/High Altitudes (1000 feet < Altitude < 2000 feet)

At altitudes between 1000 feet and 2000 feet, the turbulence velocities and turbulence angular rates are determined by linearly interpolating between the value from the low altitude model at 1000 feet transformed from mean horizontal wind coordinates to body coordinates and the value from the high altitude model at 2000 feet in body coordinates.

# Dryden Wind Turbulence Model (Continuous)

## Dialog Box



## Units

Define the units of wind speed due to the turbulence.

Units	Wind Velocity	Altitude	Airspeed
Metric (MKS)	Meters/second	Meters	Meters/second
English (Velocity in ft/s)	Feet/second	Feet	Feet/second
English (Velocity in kts)	Knots	Feet	Knots

# Dryden Wind Turbulence Model (Continuous)

---

## Specification

Define which military reference to use. This affects the application of turbulence scale lengths in the lateral and vertical directions.

## Model type

Select the wind turbulence model to use.

Continuous Von Karman (+q -r)	Use continuous representation of Von Kármán velocity spectra with positive vertical and negative lateral angular rates spectra.
Continuous Von Karman (+q +r)	Use continuous representation of Von Kármán velocity spectra with positive vertical and lateral angular rates spectra.
Continuous Von Karman (-q +r)	Use continuous representation of Von Kármán velocity spectra with negative vertical and positive lateral angular rates spectra.
Continuous Dryden (+q -r)	Use continuous representation of Dryden velocity spectra with positive vertical and negative lateral angular rates spectra.
Continuous Dryden (+q +r)	Use continuous representation of Dryden velocity spectra with positive vertical and lateral angular rates spectra.
Continuous Dryden (-q +r)	Use continuous representation of Dryden velocity spectra with negative vertical and positive lateral angular rates spectra.

# Dryden Wind Turbulence Model (Continuous)

---

Discrete Dryden (+q -r)	Use discrete representation of Dryden velocity spectra with positive vertical and negative lateral angular rates spectra.
Discrete Dryden (+q +r)	Use discrete representation of Dryden velocity spectra with positive vertical and lateral angular rates spectra.
Discrete Dryden (-q +r)	Use discrete representation of Dryden velocity spectra with negative vertical and positive lateral angular rates spectra.

The Continuous Dryden selections conform to the transfer function descriptions.

## **Wind speed at 6 m defines the low altitude intensity**

The measured wind speed at a height of 6 meters (20 feet) provides the intensity for the low-altitude turbulence model.

## **Wind direction at 6 m (degrees clockwise from north)**

The measured wind direction at a height of 6 meters (20 feet) is an angle to aid in transforming the low-altitude turbulence model into a body coordinates.

## **Probability of exceedance of high-altitude intensity**

Above 2000 feet, the turbulence intensity is determined from a lookup table that gives the turbulence intensity as a function of altitude and the probability of the turbulence intensity's being exceeded.

## **Scale length at medium/high altitudes (m)**

The turbulence scale length above 2000 feet is assumed constant, and from the military references, a figure of 1750 feet is recommended for the longitudinal turbulence scale length of the Dryden spectra.

# Dryden Wind Turbulence Model (Continuous)

---

**Note** An alternate scale length value changes the power spectral density asymptote and gust load.

---

## Wingspan

The wingspan is required in the calculation of the turbulence on the angular rates.

## Band-limited noise sample time (sec)

The sample time at which the unit variance white noise signal is generated.

## Noise seeds

There are four random numbers required to generate the turbulence signals, one for each of the three velocity components and one for the roll rate. The turbulences on the pitch and yaw angular rates are based on further shaping of the outputs from the shaping filters for the vertical and lateral velocities.

## Turbulence on

Selecting the check box generates the turbulence signals.

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the altitude, in units selected.
Second		Contains the aircraft speed, in units selected.
Third		Contains the direction cosine matrix.

Output	Dimension Type	Description
First	Three-element signal	Contains the turbulence velocities, in the selected units.
Second	Three-element signal	Contains the turbulence angular rates, in radians per second.

# Dryden Wind Turbulence Model (Continuous)

---

## Assumptions and Limitations

The frozen turbulence field assumption is valid for the cases of mean-wind velocity and the root-mean-square turbulence velocity, or intensity, is small relative to the aircraft's ground speed.

The turbulence model describes an average of all conditions for clear air turbulence because the following factors are not incorporated into the model:

- Terrain roughness
- Lapse rate
- Wind shears
- Mean wind magnitude
- Other meteorological factors (except altitude)

## Examples

See Airframe/Environment Models/Wind Models in `aeroblk_HL20` for an example of this block.

## References

U.S. Military Handbook MIL-HDBK-1797, 19 December 1997.

U.S. Military Specification MIL-F-8785C, 5 November 1980.

Chalk, C., Neal, P., Harris, T., Pritchard, F., Woodcock, R., "Background Information and User Guide for MIL-F-8785B(ASG), 'Military Specification-Flying Qualities of Piloted Airplanes'," AD869856, Cornell Aeronautical Laboratory, August 1969.

Hoblit, F., *Gust Loads on Aircraft: Concepts and Applications*, AIAA Education Series, 1988.

Ly, U., Chan, Y., "Time-Domain Computation of Aircraft Gust Covariance Matrices," AIAA Paper 80-1615, Atmospheric Flight Mechanics Conference, Danvers, Massachusetts, August 11-13, 1980.

McRuer, D., Ashkenas, I., Graham, D., *Aircraft Dynamics and Automatic Control*, Princeton University Press, July 1990.



# Dryden Wind Turbulence Model (Continuous)

---

Moorhouse, D., Woodcock, R., “Background Information and User Guide for MIL-F-8785C, ‘Military Specification-Flying Qualities of Piloted Airplanes’,” ADA119421, Flight Dynamic Laboratory, July 1982.

McFarland, R., “A Standard Kinematic Model for Flight Simulation at NASA-Ames,” NASA CR-2497, Computer Sciences Corporation, January 1975.

Tatom, F., Smith, R., Fichtl, G., “Simulation of Atmospheric Turbulent Gusts and Gust Gradients,” AIAA Paper 81-0300, Aerospace Sciences Meeting, St. Louis, Missouri, January 12-15, 1981.

Yeager, J., “Implementation and Testing of Turbulence Models for the F18-HARV Simulation,” NASA CR-1998-206937, Lockheed Martin Engineering & Sciences, March 1998.

## **See Also**

Dryden Wind Turbulence Model (Discrete)

Discrete Wind Gust Model

Wind Shear Model

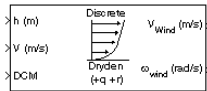
Von Karman Wind Turbulence Model (Continuous)

# Dryden Wind Turbulence Model (Discrete)

**Purpose** Generate discrete wind turbulence with Dryden velocity spectra

**Library** Environment/Wind

## Description



The Dryden Wind Turbulence Model (Discrete) block uses the Dryden spectral representation to add turbulence to the aerospace model by using band-limited white noise with appropriate digital filter finite difference equations. This block implements the mathematical representation in the Military Specification MIL-F-8785C and Military Handbook MIL-HDBK-1797.

According to the military references, turbulence is a stochastic process defined by velocity spectra. For an aircraft flying at a speed  $V$  through a frozen turbulence field with a spatial frequency of  $\Omega$  radians per meter, the circular frequency  $\omega$  is calculated by multiplying  $V$  by  $\Omega$ . The following table displays the component spectra functions:

	MIL-F-8785C	MIL-HDBK-1797
<b>Longitudinal</b>		
$\Phi_u(\omega)$	$\frac{2\sigma_u^2 L_u}{\pi V} \cdot \frac{1}{1 + \left(L_u \frac{\omega}{V}\right)^2}$	$\frac{2\sigma_u^2 L_u}{\pi V} \cdot \frac{1}{1 + \left(L_u \frac{\omega}{V}\right)^2}$
$\Phi_p(\omega)$	$\frac{\sigma_w^2}{VL_w} \cdot \frac{0.8 \left(\frac{\pi L_w}{4b}\right)^{1/3}}{1 + \left(\frac{4b\omega}{\pi V}\right)^2}$	$\frac{\sigma_w^2}{2VL_w} \cdot \frac{0.8 \left(\frac{2\pi L_w}{4b}\right)^{1/3}}{1 + \left(\frac{4b\omega}{\pi V}\right)^2}$

# Dryden Wind Turbulence Model (Discrete)

	MIL-F-8785C	MIL-HDBK-1797
<b>Lateral</b>		
$\Phi_v(\omega)$	$\frac{\sigma_v^2 L_v}{\pi V} \cdot \frac{1 + 3 \left( L_v \frac{\omega}{V} \right)^2}{\left[ 1 + \left( L_v \frac{\omega}{V} \right)^2 \right]^2}$	$\frac{2\sigma_v^2 L_v}{\pi V} \cdot \frac{1 + 12 \left( L_v \frac{\omega}{V} \right)^2}{\left[ 1 + 4 \left( L_v \frac{\omega}{V} \right)^2 \right]^2}$
$\Phi_r(\omega)$	$\frac{\mp \left( \frac{\omega}{V} \right)^2}{1 + \left( \frac{3b\omega}{\pi V} \right)^2} \cdot \Phi_v(\omega)$	$\frac{\mp \left( \frac{\omega}{V} \right)^2}{1 + \left( \frac{3b\omega}{\pi V} \right)^2} \cdot \Phi_v(\omega)$
<b>Vertical</b>		
$\Phi_w(\omega)$	$\frac{\sigma_w^2 L_w}{\pi V} \cdot \frac{1 + 3 \left( L_w \frac{\omega}{V} \right)^2}{\left[ 1 + \left( L_w \frac{\omega}{V} \right)^2 \right]^2}$	$\frac{2\sigma_w^2 L_w}{\pi V} \cdot \frac{1 + 12 \left( L_w \frac{\omega}{V} \right)^2}{\left[ 1 + 4 \left( L_w \frac{\omega}{V} \right)^2 \right]^2}$
$\Phi_q(\omega)$	$\frac{\pm \left( \frac{\omega}{V} \right)^2}{1 + \left( \frac{4b\omega}{\pi V} \right)^2} \cdot \Phi_w(\omega)$	$\frac{\pm \left( \frac{\omega}{V} \right)^2}{1 + \left( \frac{4b\omega}{\pi V} \right)^2} \cdot \Phi_w(\omega)$

# Dryden Wind Turbulence Model (Discrete)

---

The variable  $b$  represents the aircraft wingspan. The variables  $L_u$ ,  $L_v$ ,  $L_w$  represent the turbulence scale lengths. The variables  $\sigma_u$ ,  $\sigma_v$ ,  $\sigma_w$  represent the turbulence intensities.

The spectral density definitions of turbulence angular rates are defined in the references as three variations, which are displayed in the following table:

$$\begin{array}{lll} p_g = \frac{\partial w_g}{\partial y} & q_g = \frac{\partial w_g}{\partial x} & r_g = -\frac{\partial v_g}{\partial x} \\ p_g = \frac{\partial w_g}{\partial y} & q_g = \frac{\partial w_g}{\partial x} & r_g = \frac{\partial v_g}{\partial x} \\ p_g = -\frac{\partial w_g}{\partial y} & q_g = -\frac{\partial w_g}{\partial x} & r_g = \frac{\partial v_g}{\partial x} \end{array}$$

The variations affect only the vertical ( $q_g$ ) and lateral ( $r_g$ ) turbulence angular rates.

Keep in mind that the longitudinal turbulence angular rate spectrum,  $\Phi_p(\omega)$ , is a rational function. The rational function is derived from curve-fitting a complex algebraic function, not the vertical turbulence velocity spectrum,  $\Phi_w(\omega)$ , multiplied by a scale factor. Because the turbulence angular rate spectra contribute less to the aircraft gust response than the turbulence velocity spectra, it may explain the variations in their definitions.

The variations lead to the following combinations of vertical and lateral turbulence angular rate spectra:

# Dryden Wind Turbulence Model (Discrete)

## Vertical

$$\Phi_q(\omega)$$

$$\Phi_q(\omega)$$

$$-\Phi_q(\omega)$$

## Lateral

$$-\Phi_r(\omega)$$

$$\Phi_r(\omega)$$

$$\Phi_r(\omega)$$

To generate a signal with the correct characteristics, a unit variance, band-limited white noise signal is used in the digital filter finite difference equations.

The following table displays the digital filter finite difference equations:

	<b>MIL-F-8785C</b>	<b>MIL-HDBK-1797</b>
<b>Longitudinal</b>		
$u_g$	$\left(1 - \frac{V}{L_u} T\right) u_g + \sqrt{2 \frac{V}{L_u} T} \frac{\sigma_u}{\sigma_\eta} \eta_1$	$\left(1 - \frac{V}{L_u} T\right) u_g + \sqrt{2 \frac{V}{L_u} T} \frac{\sigma_u}{\sigma_\eta} \eta_1$
$p_g$	$\left(1 - \frac{2.6}{\sqrt{L_w b}} T\right) p_g +$ $\sqrt{2 \frac{2.6}{\sqrt{L_w b}} T} \frac{0.95}{\sqrt[3]{2 L_w b^2}} \frac{\sigma_w}{\sigma_\eta} \eta_4$	$\left(1 - \frac{2.6}{\sqrt{2 L_w b}} T\right) p_g +$ $\sqrt{2 \frac{2.6}{\sqrt{2 L_w b}} T} \frac{1.9}{\sqrt{2 L_w b}} \frac{\sigma_w}{\sigma_\eta} \eta_4$
<b>Lateral</b>		
$v_g$	$\left(1 - \frac{V}{L_u} T\right) v_g + \sqrt{2 \frac{V}{L_u} T} \frac{\sigma_v}{\sigma_\eta} \eta_2$	$\left(1 - \frac{V}{L_u} T\right) v_g + \sqrt{2 \frac{V}{L_u} T} \frac{\sigma_v}{\sigma_\eta} \eta_2$

# Dryden Wind Turbulence Model (Discrete)

	MIL-F-8785C	MIL-HDBK-1797
$r_g$	$\left(1 - \frac{\pi V}{3b} T\right) r_g \mp \frac{\pi}{3b} (v_g - v_{g_{past}})$	$\left(1 - \frac{\pi V}{3b} T\right) r_g \mp \frac{\pi}{3b} (v_g - v_{g_{past}})$
<b>Vertical</b>		
$w_g$	$\left(1 - \frac{V}{L_u} T\right) w_g + \sqrt{2 \frac{V}{L_u} T} \frac{\sigma_w}{\sigma_\eta} \eta_3$	$\left(1 - \frac{V}{L_u} T\right) w_g + \sqrt{2 \frac{V}{L_u} T} \frac{\sigma_w}{\sigma_\eta} \eta_3$
$q_g$	$\left(1 - \frac{\pi V}{4b} T\right) q_g \pm \frac{\pi}{4b} (w_g - w_{g_{past}})$	$\left(1 - \frac{\pi V}{4b} T\right) q_g \pm \frac{\pi}{4b} (w_g - w_{g_{past}})$

Divided into two distinct regions, the turbulence scale lengths and intensities are functions of altitude.

## Low-Altitude Model (Altitude < 1000 feet)

According to the military references, the turbulence scale lengths at low altitudes, where  $h$  is the altitude in feet, are represented in the following table:

MIL-F-8785C	MIL-HDBK-1797
$L_w = h$	$2L_w = h$
$L_u = L_v = \frac{h}{(0.177 + 0.000823h)^{1.2}}$	$L_u = 2L_v = \frac{h}{(0.177 + 0.000823h)^{1.2}}$

The turbulence intensities are given below, where  $W_{20}$  is the wind speed at 20 feet (6 m). Typically for light turbulence, the wind speed at 20 feet is 15 knots; for moderate turbulence, the wind speed is 30 knots, and for severe turbulence, the wind speed is 45 knots.

# Dryden Wind Turbulence Model (Discrete)

$$\sigma_w = 0.1W_{20}$$
$$\frac{\sigma_u}{\sigma_w} = \frac{\sigma_v}{\sigma_w} = \frac{1}{(0.177 + 0.000823h)^{0.4}}$$

The turbulence axes orientation in this region is defined as follows:

- Longitudinal turbulence velocity,  $u_g$ , aligned along the horizontal relative mean wind vector
- Vertical turbulence velocity,  $w_g$ , aligned with vertical.

At this altitude range, the output of the block is transformed into body coordinates.

## Medium/High Altitudes (Altitude > 2000 feet)

For medium to high altitudes the turbulence scale lengths and intensities are based on the assumption that the turbulence is isotropic. In the military references, the scale lengths are represented by the following equations:

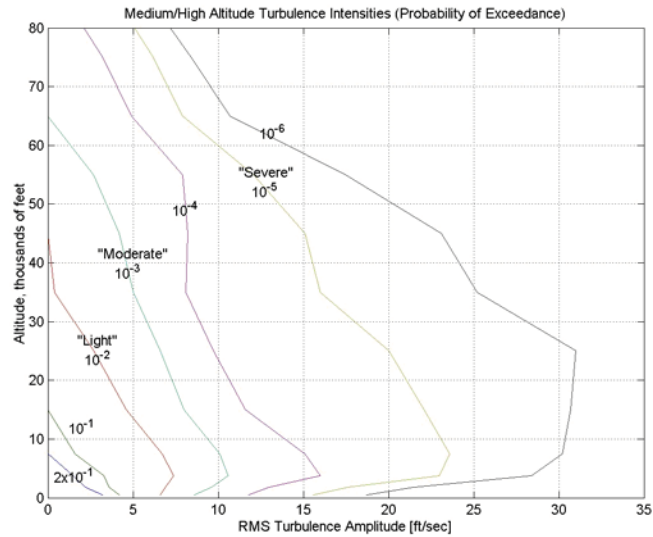
MIL-F-8785C	MIL-HDBK-1797
$L_u = L_v = L_w = 1750$ ft	$L_u = 2L_v = 2L_w = 1750$ ft

The turbulence intensities are determined from a lookup table that provides the turbulence intensity as a function of altitude and the probability of the turbulence intensity being exceeded. The relationship of the turbulence intensities is represented in the following equation:

$$\sigma_u = \sigma_v = \sigma_w.$$

The turbulence axes orientation in this region is defined as being aligned with the body coordinates.

# Dryden Wind Turbulence Model (Discrete)



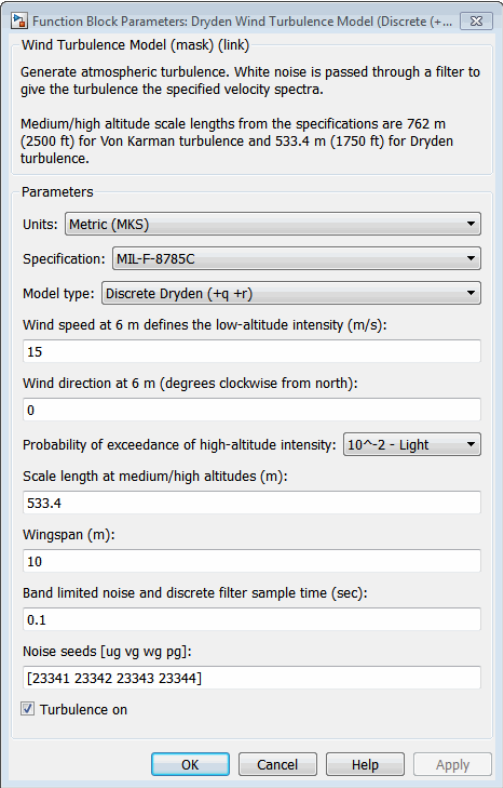
## Between Low and Medium/High Altitudes (1000 feet < Altitude < 2000 feet)

At altitudes between 1000 feet and 2000 feet, the turbulence velocities and turbulence angular rates are determined by linearly interpolating between the value from the low altitude model at 1000 feet transformed from mean horizontal wind coordinates to body coordinates and the value from the high altitude model at 2000 feet in body coordinates.



# Dryden Wind Turbulence Model (Discrete)

## Dialog Box



### Units

Define the units of wind speed due to the turbulence.

# Dryden Wind Turbulence Model (Discrete)

---

<b>Units</b>	<b>Wind Velocity</b>	<b>Altitude</b>	<b>Airspeed</b>
Metric (MKS)	Meters/second	Meters	Meters/second
English (Velocity in ft/s)	Feet/second	Feet	Feet/second
English (Velocity in kts)	Knots	Feet	Knots

## Specification

Define which military reference to use. This affects the application of turbulence scale lengths in the lateral and vertical directions

## Model type

Select the wind turbulence model to use:

Continuous Von Karman (+q -r)	Use continuous representation of Von Kármán velocity spectra with positive vertical and negative lateral angular rates spectra.
Continuous Von Karman (+q +r)	Use continuous representation of Von Kármán velocity spectra with positive vertical and lateral angular rates spectra.
Continuous Von Karman (-q +r)	Use continuous representation of Von Kármán velocity spectra with negative vertical and positive lateral angular rates spectra.
Continuous Dryden (+q -r)	Use continuous representation of Dryden velocity spectra with positive vertical and negative lateral angular rates spectra.

# Dryden Wind Turbulence Model (Discrete)

---

Continuous Dryden (+q +r)	Use continuous representation of Dryden velocity spectra with positive vertical and lateral angular rates spectra.
Continuous Dryden (-q +r)	Use continuous representation of Dryden velocity spectra with negative vertical and positive lateral angular rates spectra.
Discrete Dryden (+q -r)	Use discrete representation of Dryden velocity spectra with positive vertical and negative lateral angular rates spectra.
Discrete Dryden (+q +r)	Use discrete representation of Dryden velocity spectra with positive vertical and lateral angular rates spectra.
Discrete Dryden (-q +r)	Use discrete representation of Dryden velocity spectra with negative vertical and positive lateral angular rates spectra.

The Discrete Dryden selections conform to the transfer function descriptions.

## **Wind speed at 6 m defines the low altitude intensity**

The measured wind speed at a height of 6 meters (20 feet) provides the intensity for the low-altitude turbulence model.

## **Wind direction at 6 m (degrees clockwise from north)**

The measured wind direction at a height of 6 meters (20 feet) is an angle to aid in transforming the low-altitude turbulence model into a body coordinates.

## **Probability of exceedance of high-altitude intensity**

Above 2000 feet, the turbulence intensity is determined from a lookup table that gives the turbulence intensity as a function of

# Dryden Wind Turbulence Model (Discrete)

---

altitude and the probability of the turbulence intensity's being exceeded.

## **Scale length at medium/high altitudes**

The turbulence scale length above 2000 feet is assumed constant, and from the military references, a figure of 1750 feet is recommended for the longitudinal turbulence scale length of the Dryden spectra.

---

**Note** An alternate scale length value changes the power spectral density asymptote and gust load.

---

## **Wingspan**

The wingspan is required in the calculation of the turbulence on the angular rates.

## **Band-limited noise and discrete filter sample time (sec)**

The sample time at which the unit variance white noise signal is generated and at which the discrete filters are updated.

## **Noise seeds**

There are four random numbers required to generate the turbulence signals, one for each of the three velocity components and one for the roll rate. The turbulences on the pitch and yaw angular rates are based on further shaping of the outputs from the shaping filters for the vertical and lateral velocities.

## **Turbulence on**

Selecting the check box generates the turbulence signals.

# Dryden Wind Turbulence Model (Discrete)

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the altitude, in units selected.
Second		Contains the aircraft speed, in units selected.
Third		Contains the NED direction cosine matrix.

Output	Dimension Type	Description
First	Three-element signal	Contains the turbulence velocities, in the selected units.
Second	Three-element signal	Contains the turbulence angular rates, in radians per second.

## Assumptions and Limitations

The “frozen turbulence field” assumption is valid for the cases of mean-wind velocity and the root-mean-square turbulence velocity, or intensity, is small relative to the aircraft’s ground speed.

The turbulence model describes an average of all conditions for clear air turbulence because the following factors are not incorporated into the model:

- Terrain roughness
- Lapse rate
- Wind shears
- Mean wind magnitude
- Other meteorological factors (except altitude)

## References

- U.S. Military Handbook MIL-HDBK-1797, 19 December 1997.  
U.S. Military Specification MIL-F-8785C, 5 November 1980.

# Dryden Wind Turbulence Model (Discrete)

---

Chalk, C., Neal, P., Harris, T., Pritchard, F., Woodcock, R., "Background Information and User Guide for MIL-F-8785B(ASG), 'Military Specification-Flying Qualities of Piloted Airplanes'," AD869856, Cornell Aeronautical Laboratory, August 1969.

Hoblitt, F., *Gust Loads on Aircraft: Concepts and Applications*, AIAA Education Series, 1988.

Ly, U., Chan, Y., "Time-Domain Computation of Aircraft Gust Covariance Matrices," AIAA Paper 80-1615, Atmospheric Flight Mechanics Conference, Danvers, Massachusetts, August 11-13, 1980.

McRuer, D., Ashkenas, I., Graham, D., *Aircraft Dynamics and Automatic Control*, Princeton University Press, July 1990.

Moorhouse, D., Woodcock, R., "Background Information and User Guide for MIL-F-8785C, 'Military Specification-Flying Qualities of Piloted Airplanes'," ADA119421, Flight Dynamic Laboratory, July 1982.

McFarland, R., "A Standard Kinematic Model for Flight Simulation at NASA-Ames," NASA CR-2497, Computer Sciences Corporation, January 1975.

Tatom, F., Smith, R., Fichtl, G., "Simulation of Atmospheric Turbulent Gusts and Gust Gradients," AIAA Paper 81-0300, Aerospace Sciences Meeting, St. Louis, Missouri, January 12-15, 1981.

Yeager, J., "Implementation and Testing of Turbulence Models for the F18-HARV Simulation," NASA CR-1998-206937, Lockheed Martin Engineering & Sciences, March 1998.

## See Also

Dryden Wind Turbulence Model (Continuous)

Von Karman Wind Turbulence Model (Continuous)

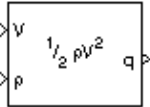
Discrete Wind Gust Model

Wind Shear Model

**Purpose** Compute dynamic pressure using velocity and air density

**Library** Flight Parameters

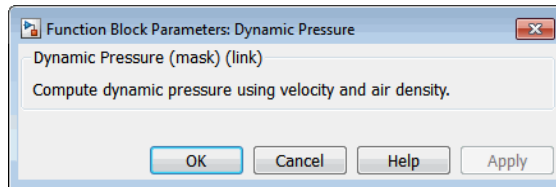
**Description** The Dynamic Pressure block computes dynamic pressure. Dynamic pressure is defined as



$$\bar{q} = \frac{1}{2} \rho V^2$$

where  $\rho$  is air density and  $V$  is velocity.

## Dialog Box



## Inputs and Outputs

Input	Dimension Type	Description
First	Vector	Contains the velocity.
Second		Contains the air density.

Output	Dimension Type	Description
First		Contains the dynamic pressure.

**Examples** See the Airframe subsystem in aeroblk\_HL20 for an example of this block.

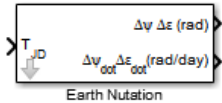
**See Also** Aerodynamic Forces and Moments  
Mach Number

# Earth Nutation

**Purpose** Implement Earth nutation

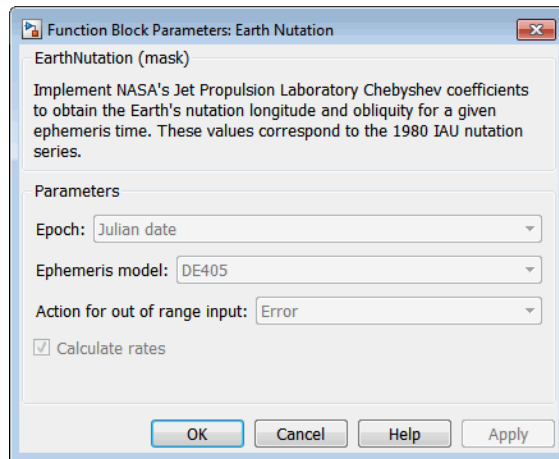
**Library** Environment/Celestial Phenomena

**Description** For a given Julian date, the Earth Nutation block implements the International Astronomical Union (IAU) 1980 nutation series. The block uses the Chebyshev coefficients that the NASA Jet Propulsion Laboratory provides.



**Tip** As input for the block, you can use the Aerospace Toolbox `juliandate` function to calculate the Julian date, or calculate your own Julian date, and input them using the Constant block.

## Dialog Box



### Epoch

Select one of the following:

- Julian date

Julian date to calculate the Earth nutation. When this option is selected, the block has one input port.



- $T_0$  and elapsed Julian time

Julian date, specified by two block inputs:

- A fixed starting epoch ( $T_0$ ).
- Variable elapsed time between  $T_0$  and the desired model simulation time.

$T_0$  plus the variable elapsed time cannot exceed the maximum Julian date for the specified **Ephemerides**.

### Ephemeris model

Select one of the following ephemerides models defined by the Jet Propulsion Laboratory.

Ephemerides Model	Description
DE405	<p>Released in 1998. This ephemerides takes into account the Julian date range 2305424.50 (December 9, 1599 ) to 2525008.50 (February 20, 2201).</p> <p>This block implements these ephemerides with respect to the International Celestial Reference Frame version 1.0, adopted in 1998.</p>
DE421	<p>Released in 2008. This ephemerides takes into account the Julian date range 2414992.5 (December 4, 1899) to 2469808.5 (January 2, 2050).</p> <p>This block implements these ephemerides with respect to the International Celestial Reference Frame version 1.0, adopted in 1998.</p>
DE423	<p>Released in 2010. This ephemerides takes into account the Julian date range 2378480.5 (December 16, 1799) to 2524624.5 (February 1, 2200).</p> <p>This block implements these ephemerides with respect to the International Celestial Reference Frame version 2.0, adopted in 2010.</p>

# Earth Nutation

## Action for out of range input

Specify the block behavior when the block inputs are out of range.

Action	Description
None	No action.
Warning	Warning in the MATLAB Command Window, model simulation continues.
Error (default)	MATLAB returns an exception, model simulation stops.

## Calculate rates

Select this check box to calculate the rate of the Earth nutation and add a second block output.

## Inputs and Outputs

Input	Dimension Type	Description
First	Scalar	<ul style="list-style-type: none"><li>Julian date (<math>T_{JD}</math>) (default) — One input. Specify a date between the minimum and maximum Julian date.</li><li>Fixed Julian date (<math>T0_{JD}</math>) plus the elapsed Julian time (<math>\Delta T_{JD}</math>) between the fixed date and the ephemeris time. — Two inputs, where the first input is <math>T0_{JD}</math> and the second input is <math>\Delta T_{JD}</math>. <math>\Delta T_{JD}</math> must be a positive number. The sum of <math>T0_{JD}</math> and <math>\Delta T_{JD}</math> must fall between the minimum and maximum Julian date.</li></ul> <p>The block <b>Epoch</b> parameter controls the number of block inputs.</p>

Input	Dimension Type	Description
Second (Optional)	Scalar	<p>See the <b>Ephemerides</b> parameter for the minimum and maximum Julian dates.</p> <p><math>\Delta T_{JD}</math> — Elapsed Julian time (<math>\Delta T_{JD}</math>) between the fixed date and the ephemeris time. The sum of <math>T0_{JD}</math> and <math>\Delta T_{JD}</math> must fall between the minimum and maximum Julian date.</p> <p>See the <b>Ephemerides</b> parameter for the minimum and maximum Julian date.</p>
Output	Dimension Type	Description
First	Vector	Earth nutation in longitude ( $\Delta\psi$ ) and obliquity ( $\Delta\epsilon$ ). Units are radians.
Second (Optional)	Vector	Earth nutation angular rate for the longitude ( $\Delta\psi_{\text{dot}}$ ) and obliquity ( $\Delta\epsilon_{\text{dot}}$ ). Units are radians/day.

## References

Folkner, W. M., J. G. Williams, D. H. Boggs, “The Planetary and Lunar Ephemeris DE 421,” *IPN Progress Report 42-178*, 2009.

Vallado, D. A., *Fundamentals of Astrodynamics and Applications*, McGraw-Hill, New York, 1997.

## See Also

Moon Libration | Planetary Ephemeris |

## External Web Sites

- [http://ssd.jpl.nasa.gov/?planet\\_eph\\_export](http://ssd.jpl.nasa.gov/?planet_eph_export)

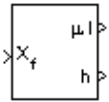
# ECEF Position to LLA

---

**Purpose** Calculate geodetic latitude, longitude, and altitude above planetary ellipsoid from Earth-centered Earth-fixed (ECEF) position

**Library** Utilities/Axes Transformations

**Description** The ECEF Position to LLA block converts a 3-by-1 vector of ECEF position ( $\bar{p}$ ) into geodetic latitude ( $\bar{\mu}$ ), longitude ( $\bar{\iota}$ ), and altitude ( $\bar{h}$ ) above the planetary ellipsoid.



The ECEF position is defined as

$$\bar{p} = \begin{bmatrix} \bar{p}_x \\ \bar{p}_y \\ \bar{p}_z \end{bmatrix}$$

Longitude is calculated from the ECEF position by

$$\iota = \text{atan} \left( \frac{p_y}{p_x} \right)$$

Geodetic latitude ( $\bar{\mu}$ ) is calculated from the ECEF position using Bowring's method, which typically converges after two or three iterations. The method begins with an initial guess for geodetic latitude ( $\bar{\mu}$ ) and reduced latitude ( $\bar{\beta}$ ). An initial guess takes the form:

$$\bar{\beta} = \text{atan} \left( \frac{p_z}{(1-f)s} \right)$$

$$\bar{\mu} = \text{atan} \left( \frac{p_z + \frac{e^2(1-f)}{(1-e^2)} R(\sin \beta)^3}{s - e^2 R(\cos \beta)^3} \right)$$

where  $R$  is the equatorial radius,  $f$  the flattening of the planet,  $e^2 = 1 - (1-f)^2$ , the square of first eccentricity, and

$$s = \sqrt{p_x^2 + p_y^2}$$

After the initial guesses are calculated, the reduced latitude ( $\bar{\beta}$ ) is recalculated using

$$\beta = \text{atan} \left( \frac{(1-f) \sin \mu}{\cos \mu} \right)$$

and geodetic latitude ( $\bar{\mu}$ ) is reevaluated. This last step is repeated until  $\bar{\mu}$  converges.

The altitude ( $\bar{h}$ ) above the planetary ellipsoid is calculated with

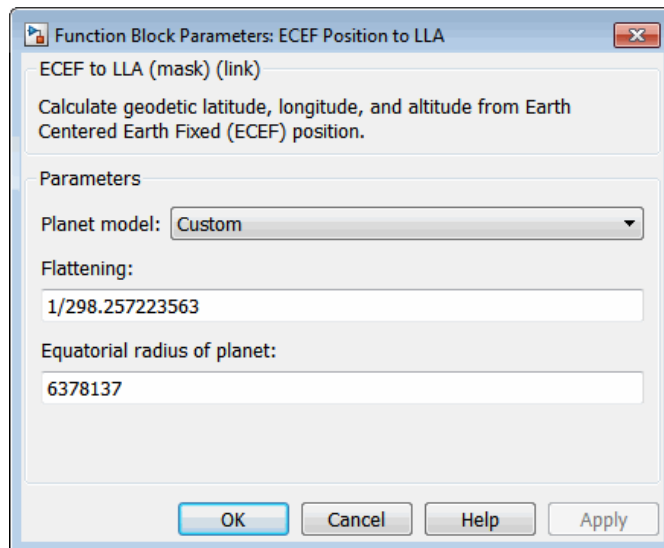
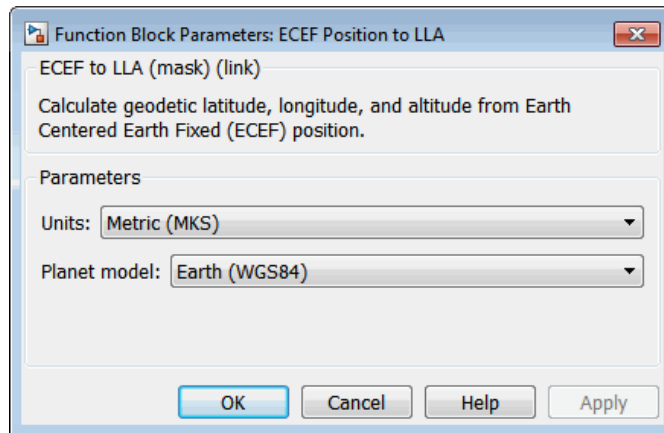
$$h = s \cos \mu + (p_z + e^2 N \sin \mu) \sin \mu - N$$

where the radius of curvature in the vertical prime ( $\bar{N}$ ) is given by

$$N = \frac{R}{\sqrt{1 - e^2 (\sin \mu)^2}}$$

# ECEF Position to LLA

## Dialog Box



## Units

Specifies the parameter and output units:

<b>Units</b>	<b>Position</b>	<b>Equatorial Radius</b>	<b>Altitude</b>
Metric (MKS)	Meters	Meters	Meters
English	Feet	Feet	Feet

This option is only available when **Planet model** is set to Earth (WGS84).

### **Planet model**

Specifies the planet model to use, Custom or Earth (WGS84).

### **Flattening**

Specifies the flattening of the planet.

This option is available only with **Planet model** set to Custom.

### **Equatorial radius of planet**

Specifies the radius of the planet at its equator. The equatorial radius units should be the same as the desired units for ECEF position.

This option is available only with **Planet model** set to Custom.

## **Inputs and Outputs**

<b>Input</b>	<b>Dimension Type</b>	<b>Description</b>
First	3-by-1 vector	Contains the position in ECEF frame.

<b>Output</b>	<b>Dimension Type</b>	<b>Description</b>
First	2-by-1 vector	Contains the geodetic latitude and longitude, in degrees.
Second	Scalar	Contains the altitude above the planetary ellipsoid, in the same units as the ECEF position.

# ECEF Position to LLA

---

## Assumptions and Limitations

This implementation generates a geodetic latitude that lies between  $\pm 90$  degrees, and longitude that lies between  $\pm 180$  degrees. The planet is assumed to be ellipsoidal. By setting the flattening to 0, you model a spherical planet.

The implementation of the ECEF coordinate system assumes that its origin lies at the center of the planet, the  $x$ -axis intersects the prime (Greenwich) meridian and the equator, the  $z$ -axis is the mean spin axis of the planet (positive to the north), and the  $y$ -axis completes the right-handed system.

## References

Stevens, B. L., and F. L. Lewis, *Aircraft Control and Simulation*, John Wiley & Sons, New York, 1992.

Zipfel, P. H., *Modeling and Simulation of Aerospace Vehicle Dynamics*, AIAA Education Series, Reston, Virginia, 2000.

“Atmospheric and Space Flight Vehicle Coordinate Systems,” ANSI/AIAA R-004-1992.

## See Also

See “About Aerospace Coordinate Systems” on page 2-12.

Direction Cosine Matrix ECEF to NED

Direction Cosine Matrix ECEF to NED to Latitude and Longitude

Geocentric to Geodetic Latitude

LLA to ECEF Position

Radius at Geocentric Latitude



## Purpose

Calculate geoid height as determined from EGM96 Geopotential Model

---

**Note** EGM96 Geoid will be removed in a future version. Use Geoid Height instead.

---

## Library

Environment/Gravity

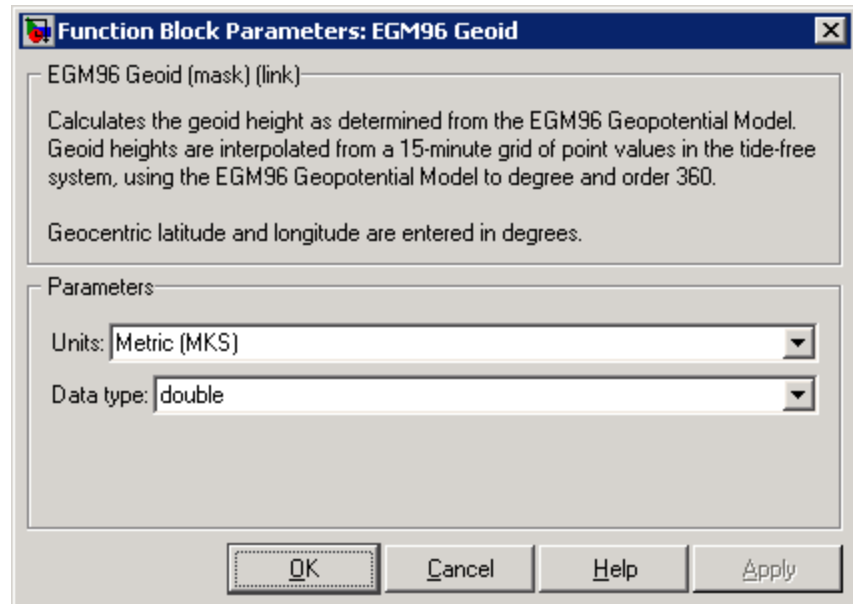
## Description



The EGM96 Geoid block calculates the geoid height as determined from the EGM96 Geopotential Model. The block interpolates the geoid heights from a 15 minute grid of point values in the tide-free system. It uses the EGM96 Geopotential Model to degree and order 360. The geoid undulations are with respect to the WGS84 ellipsoid.

The interpolation scheme wraps over the poles to allow for geoid height calculations at and near these locations.

## Dialog Box



# EGM96 Geoid

---

## Units

Specifies the parameter and output units:

Units	Height
Metric (MKS)	Meters
English	Feet

## Data type

Specify the data type of the input and output signals. From the list, select `double` or `single`.

## Inputs and Outputs

Input	Dimension Type	Description
First	Scalar	Contains geocentric latitude in degrees, where north latitude is positive, and south latitude is negative. Input latitude must be of type <code>single</code> or <code>double</code> . If latitude is not in the range from -90 to 90, the block wraps it to be within the range.
Second	Scalar	Contains geocentric longitude in degrees, where east longitude is positive in the range from 0 to 360. Input longitude must be of type <code>single</code> or <code>double</code> . If longitude is not in the range from 0 to 360, the block wraps it to be within the range.

Output	Dimension Type	Description
First	Scalar	Contains the geoid height, in meters.

## Limitations

This block has the limitations of the 1996 Earth Geopotential Model. For more information, see <http://earth-info.nga.mil/GandG/wgs84/gravitymod/egm96/egm96.html>. The WGS84 EGM96 geoid undulations have an error range of +/-0.5 to +/-1.0 meters worldwide.

## References

“Department of Defense World Geodetic System 1984, Its Definition and Relationship with Local Geodetic Systems”, NIMA TR8350.2.

“The Development of the Joint NASA GSFC and NIMA Geopotential Model EGM96”, NASA/TP-1998-206861.

National Geospatial-Intelligence Agency Web site:  
<http://earth-info.nga.mil/GandG/wgs84/gravitymod/egm96/egm96.html>

## See Also

WGS84 Gravity Model

# Estimate Center of Gravity

**Purpose** Calculate center of gravity location

**Library** Mass Properties

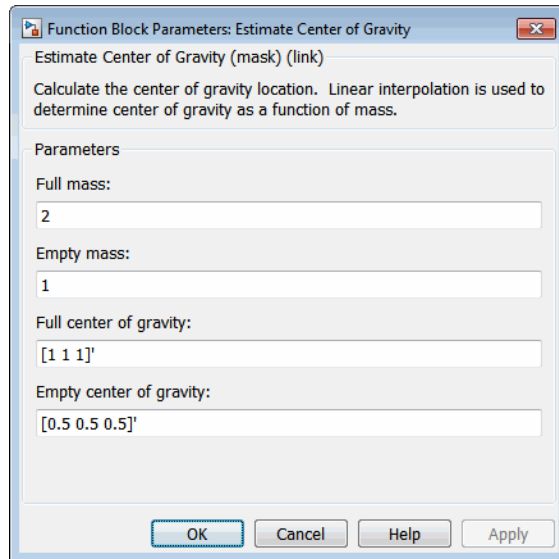
## Description



The Estimate Center of Gravity block calculates the center of gravity location and the rate of change of the center of gravity.

Linear interpolation is used to estimate the location of center of gravity as a function of mass. The rate of change of center of gravity is a linear function of rate of change of mass.

## Dialog Box



**Full mass**  
Specifies the gross mass of the craft.

**Empty mass**  
Specifies the empty mass of the craft.

**Full center of gravity**  
Specifies the center of gravity at gross mass of the craft.

## Empty center of gravity

Specifies the center of gravity at empty mass of the craft.

### Inputs and Outputs

Input	Dimension Type	Description
First		Contains the mass.
Second		Contains the rate of change of mass.

Output	Dimension Type	Description
First		Contains the center of gravity location.
Second		Contains the rate of change of center of gravity location.

### See Also

Aerodynamic Forces and Moments

Estimate Inertia Tensor

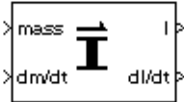
Moments About CG Due to Forces

# Estimate Inertia Tensor

**Purpose** Calculate inertia tensor

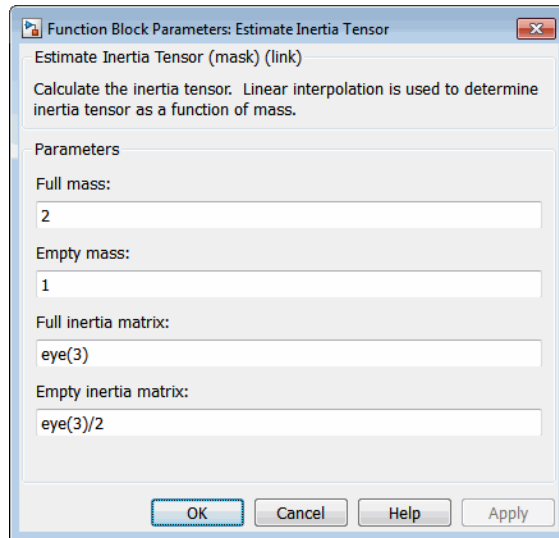
**Library** Mass Properties

**Description** The Estimate Inertia Tensor block calculates the inertia tensor and the rate of change of the inertia tensor.



Linear interpolation is used to estimate the inertia tensor as a function of mass. The rate of change of the inertia tensor is a linear function of rate of change of mass.

## Dialog Box



**Full mass** Specifies the gross mass of the craft.

**Empty mass** Specifies the empty mass of the craft.

**Full inertia matrix** Specifies the inertia tensor at gross mass of the craft.

## Empty inertia matrix

Specifies the inertia tensor at empty mass of the craft.

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the mass.
Second		Contains the rate of change of mass.

Output	Dimension Type	Description
First		Contains the inertia tensor.
Second		Contains the rate of change of inertia tensor.

## See Also

Estimate Center of Gravity

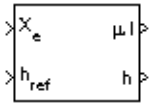
Symmetric Inertia Tensor

# Flat Earth to LLA

**Purpose** Estimate geodetic latitude, longitude, and altitude from flat Earth position

**Library** Utilities/Axes Transformations

**Description** The Flat Earth to LLA block converts a 3-by-1 vector of Flat Earth position ( $\bar{p}$ ) into geodetic latitude ( $\bar{\mu}$ ), longitude ( $\bar{\tau}$ ), and altitude ( $h$ ). The flat Earth coordinate system assumes the  $z$ -axis is downward positive. The estimation begins by transforming the flat Earth  $x$  and  $y$  coordinates to North and East coordinates. The transformation has the form of



$$\begin{bmatrix} N \\ E \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} p_x \\ p_y \end{bmatrix}$$

where ( $\bar{\psi}$ ) is the angle in degrees clockwise between the  $x$ -axis and north.

To convert the North and East coordinates to geodetic latitude and longitude, the radius of curvature in the prime vertical ( $R_N$ ) and the radius of curvature in the meridian ( $R_M$ ) are used. ( $R_N$ ) and ( $R_M$ ) are defined by the following relationships:

$$R_N = \frac{R}{\sqrt{1 - (2f - f^2)\sin^2\mu_0}}$$
$$R_M = R_N \frac{1 - (2f - f^2)}{1 - (2f - f^2)\sin^2\mu_0}$$

where ( $R$ ) is the equatorial radius of the planet and ( $\bar{f}$ ) is the flattening of the planet.

Small changes in the in latitude and longitude are approximated from small changes in the North and East positions by



$$d\mu = \operatorname{atan}\left(\frac{1}{R_M}\right)dN$$

$$d\iota = \operatorname{atan}\left(\frac{1}{R_N \cos \mu}\right)dE$$

The output latitude and longitude are simply the initial latitude and longitude plus the small changes in latitude and longitude.

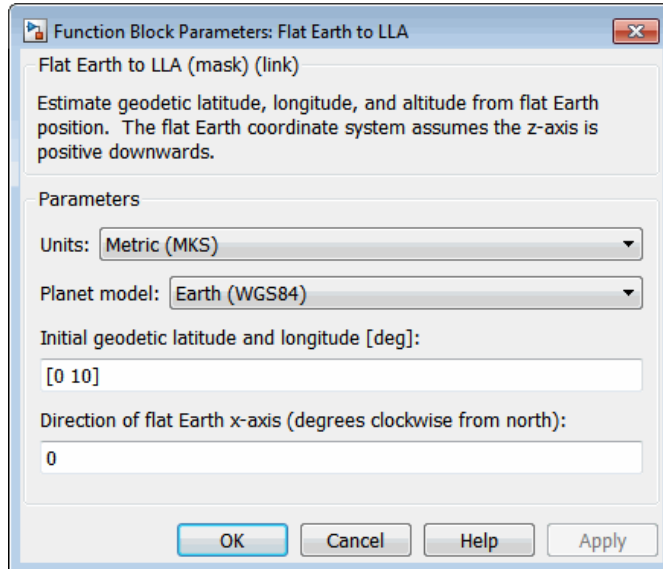
$$\mu = \mu_0 + d\mu$$

$$\iota = \iota_0 + d\iota$$

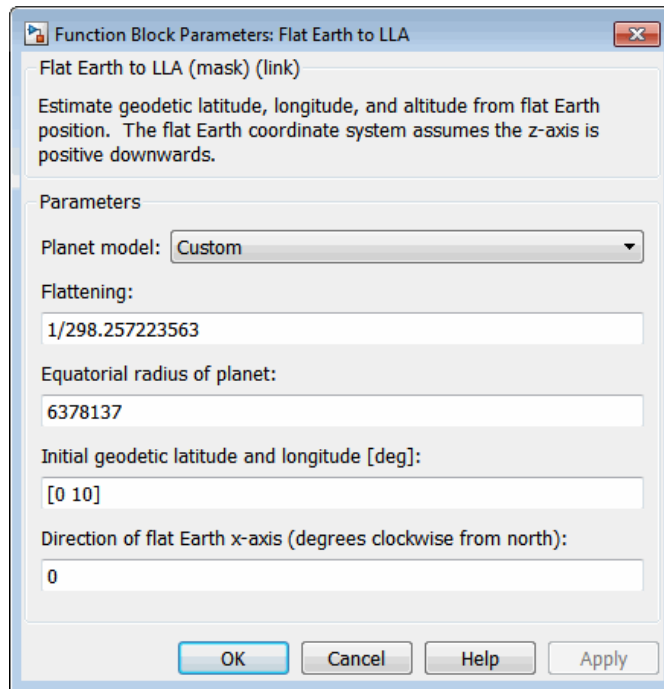
The altitude is the negative flat Earth  $z$ -axis value minus the reference height ( $h_{ref}$ ).

$$h = -p_z - h_{ref}$$

## Dialog Box



# Flat Earth to LLA



## Units

Specifies the parameter and output units:

<b>Units</b>	<b>Position</b>	<b>Equatorial Radius</b>	<b>Altitude</b>
Metric (MKS)	Meters	Meters	Meters
English	Feet	Feet	Feet

This option is only available when **Planet model** is set to Earth (WGS84).

## Planet model

Specifies the planet model to use: Custom or Earth (WGS84).

## Flattening

Specifies the flattening of the planet. This option is only available with **Planet model Custom**.

## Equatorial radius of planet

Specifies the radius of the planet at its equator. The units of the equatorial radius parameter should be the same as the units for flat Earth position. This option is only available with **Planet model Custom**.

## Initial geodetic latitude and longitude

Specifies the reference location, in degrees of latitude and longitude, for the origin of the estimation and the origin of the flat Earth coordinate system.

## Direction of flat Earth x-axis (degrees clockwise from north)

Specifies angle used for converting flat Earth x and y coordinates to North and East coordinates.

## Inputs and Outputs

Input	Dimension Type	Description
First	3-by-1 vector	Contains the position in flat Earth frame.
Second	Scalar	Contains the reference height from surface of Earth to flat Earth frame with regard to Earth frame, in same units as flat Earth position.

Output	Dimension Type	Description
First	2-by-1 vector	Contains the geodetic latitude and longitude, in degrees.
Second	Scalar	Contains the altitude above the input reference altitude, in same units as flat Earth position.

# Flat Earth to LLA

---

## Assumptions and Limitations

This estimation method assumes the flight path and bank angle are zero.

This estimation method assumes the flat Earth  $z$ -axis is normal to the Earth at the initial geodetic latitude and longitude only. This method has higher accuracy over small distances from the initial geodetic latitude and longitude, and nearer to the equator. The longitude will have higher accuracy the smaller the variations in latitude. Additionally, longitude is singular at the poles.

## References

Etkin, B., *Dynamics of Atmospheric Flight*, John Wiley & Sons, New York, 1972.

Stevens, B. L., and F. L. Lewis, *Aircraft Control and Simulation*, Second Edition, John Wiley & Sons, New York, 2003.

## See Also

Direction Cosine Matrix ECEF to NED

Direction Cosine Matrix ECEF to NED to Latitude and Longitude

ECEF Position to LLA

Geocentric to Geodetic Latitude

LLA to ECEF Position

Radius at Geocentric Latitude

# FlightGear Preconfigured 6DoF Animation

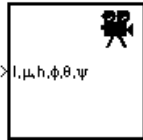
## Purpose

Connect model to FlightGear flight simulator

## Library

Animation/Flight Simulator Interfaces

## Description



The FlightGear Preconfigured 6DoF Animation block lets you drive position and attitude values to a FlightGear flight simulator vehicle given double-precision values for longitude ( $l$ ), latitude ( $\mu$ ), altitude ( $h$ ), roll ( $\phi$ ), pitch ( $\theta$ ), and yaw ( $\psi$ ), respectively.

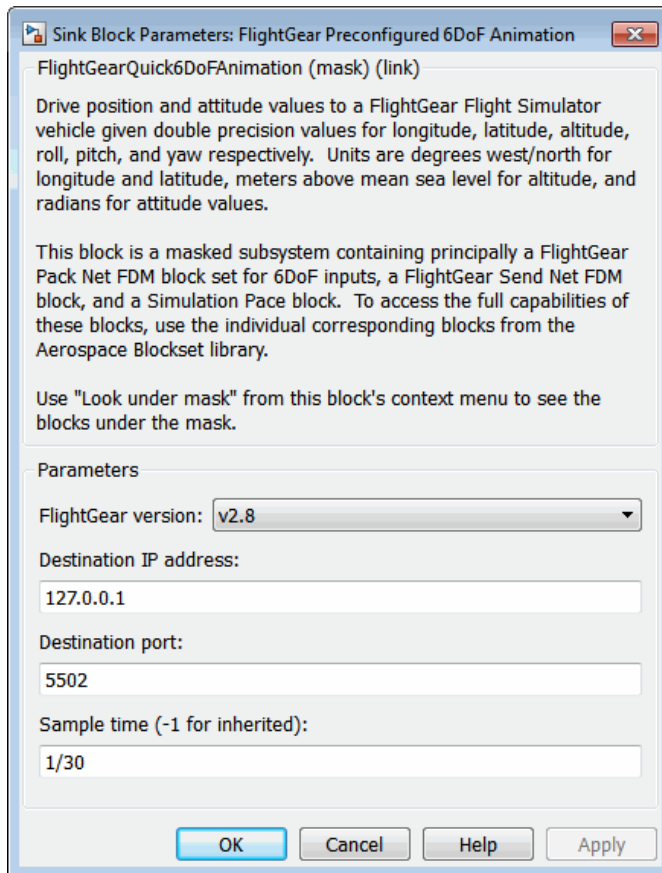
The block is a masked subsystem containing principally a Pack net\_fdm Packet for FlightGear block set for 6DoF inputs, a Send net\_fdm Packet to FlightGear block, and a Simulation Pace block. To access the full capabilities of these blocks, use the individual corresponding blocks from the Aerospace Blockset library.

The block is additionally configured as a SimViewingDevice, so that if you generate code for your model using the Simulink Coder product and connect to the running target code using the Simulink Coder External Mode available from the model toolbar, then the Simulink software can obtain the data from the target on the fly and transmit position and attitude data to FlightGear. For more information, see “Sim Viewing Devices in External Mode”.

This block does not produce deployable code, but it can be used with Simulink Coder external mode as a SimViewingDevice.

# FlightGear Preconfigured 6DoF Animation

## Dialog Box



### FlightGear version

Select your FlightGear software version.

Supported versions: v0.9.3, v0.9.8/0.9.8a, v0.9.9, v0.9.10, v1.0, v1.9.1, v2.0, v2.4, v2.6, v2.8.

### Destination IP address

Specify your destination IP address.

# FlightGear Preconfigured 6DoF Animation

---

## Destination port

Specify your destination port.

## Sample time

Specify the sample time (-1 for inherited).

## Inputs and Outputs

Input	Dimension Type	Description
First	Vector	Contains the longitude, latitude, altitude, roll, pitch, and yaw, in double-precision. Units are degrees west/north for longitude and latitude, meters above mean sea level for altitude, and radians for attitude values.

## Reference

Bowditch, N., *American Practical Navigator, An Epitome of Navigation*, US Navy Hydrographic Office, 1802.

## See Also

Generate Run Script

Pack net\_fdm Packet for FlightGear

Send net\_fdm Packet to FlightGear

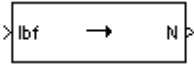
Simulation Pace

# Force Conversion

**Purpose** Convert from force units to desired force units

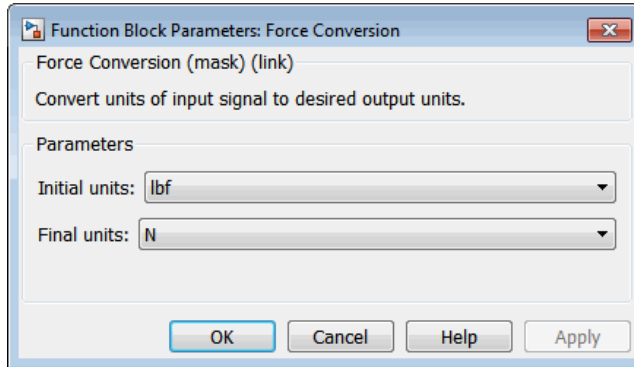
**Library** Utilities/Unit Conversions

**Description** The Force Conversion block computes the conversion factor from specified input force units to specified output force units and applies the conversion factor to the input signal.



The Force Conversion block icon displays the input and output units selected from the **Initial units** and the **Final units** lists.

## Dialog Box



**Initial units**  
Specifies the input units.

**Final units**  
Specifies the output units.

The following conversion units are available:

lbf                      Pound force

N                         Newtons



## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the force in initial force units.

Output	Dimension Type	Description
First		Contains the force in final force units.

## See Also

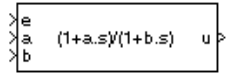
Acceleration Conversion  
Angle Conversion  
Angular Acceleration Conversion  
Angular Velocity Conversion  
Density Conversion  
Length Conversion  
Mass Conversion  
Pressure Conversion  
Temperature Conversion  
Velocity Conversion

# Gain Scheduled Lead-Lag

**Purpose** Implement first-order lead-lag with gain-scheduled coefficients

**Library** GNC/Controls

**Description** The Gain Scheduled Lead-Lag block implements a first-order lag of the form

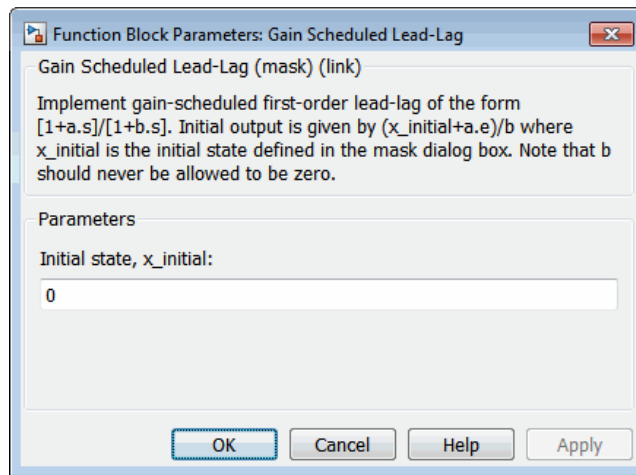


$$u = \frac{1 + as}{1 + bs} e$$

where  $e$  is the filter input, and  $u$  the filter output.

The coefficients  $a$  and  $b$  are inputs to the block, and hence can be made dependent on flight condition or operating point. For example, they could be produced from the Lookup Table (n-D) Simulink block.

## Dialog Box



### Initial state, $x_{\text{initial}}$

The initial internal state for the filter  $x_{\text{initial}}$ . Given this initial state, the initial output is given by

$$u|_{t=0} = \frac{x\_initial + ae}{b}$$

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the filter input.
Second		Contains the numerator coefficient.
Third		Contains the denominator coefficient.

Output	Dimension Type	Description
First		Contains the filter output.

# Generate Run Script

---

**Purpose** Generate FlightGear run script on current platform

**Library** Animation/Flight Simulator Interfaces

**Description** The Generate Run Script block generates a customized FlightGear run script on the current platform.



To generate the run script, fill in the required information in the dialog box fields, then click **Generate Script**.

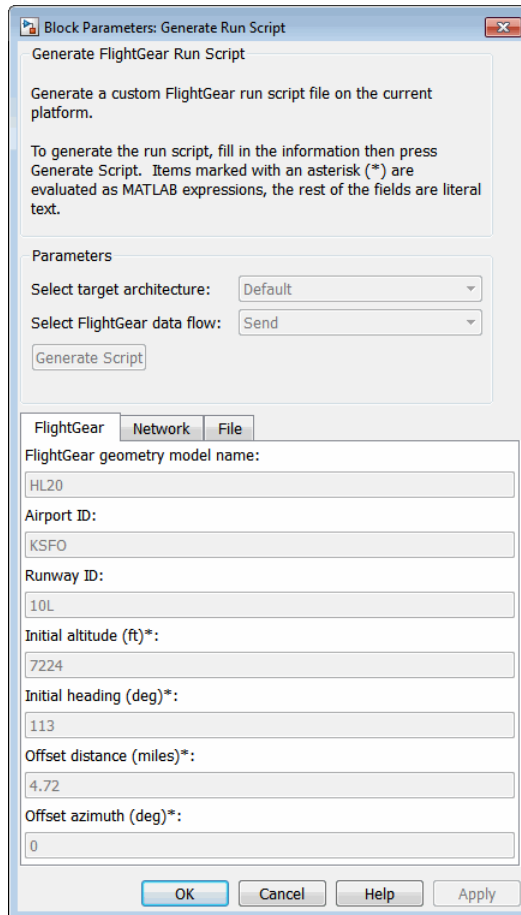
In the dialog box, fields marked with an asterisk (\*) are evaluated as MATLAB expressions. The other fields are treated as literal text.

### **For More Information About FlightGear**

See “Create a FlightGear Run Script” on page 2-36.

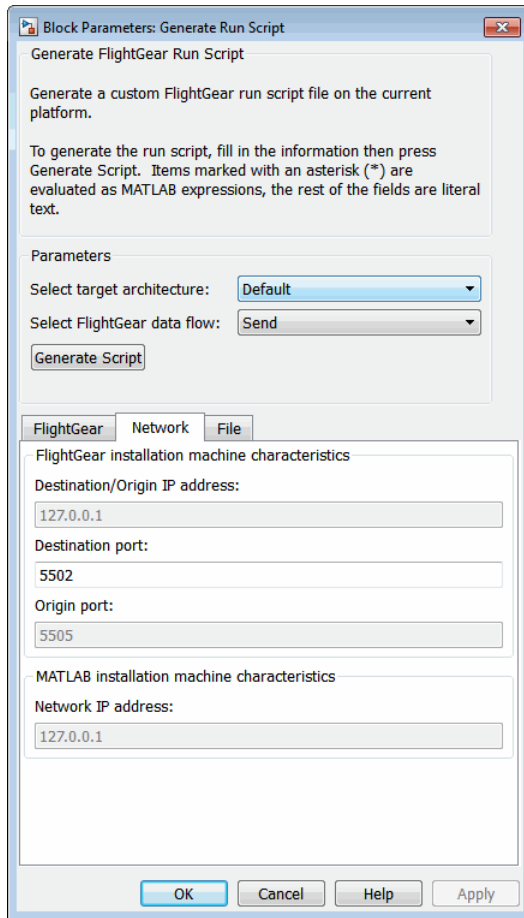
## Dialog Box

This figure is the **FlightGear** pane of the Generate Run Script block dialog box;

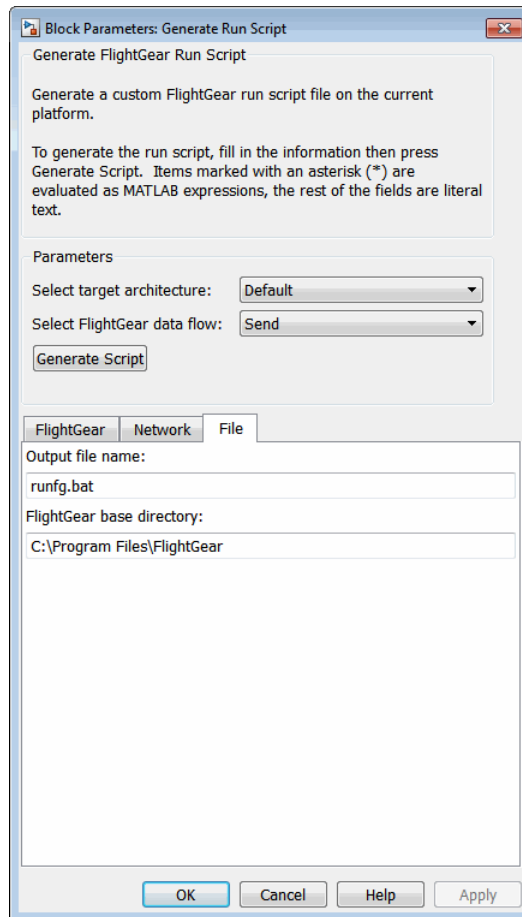


Here is the **Network** pane of the Generate Run Script block dialog box:

# Generate Run Script



The following figure is the **File** pane of the Generate Run Script block dialog box:



## Select target architecture

From the list, select the target platform on which you want to execute the run script. This platform can differ from the platform on which you create the run script. Select **Default** if you want to generate a run script to run on the platform from which you create the run script.

- Win32

# Generate Run Script

---

- Win64
- Linux
- Mac

## Select FlightGear data flow

From the list, select the direction of the data flow:

- Send  
Creates the run script to set up the sending of the `net_fdm` control model from Simulink to FlightGear.
- Receive  
Creates the run script to set up the receiving of the `net_ctrl` control model from FlightGear to Simulink.
- Send-Receive  
Creates the run script to set up FlightGear to receive and broadcast data to and from Simulink.

---

**Note** Selecting this option does not mean that you receive the same data that you sent (for example, with the **Send-Receive** option, you might not see control surface position data). With this option, you see primarily user input (such as joystick) and environmental data.

---

## FlightGear geometry model name

Specify the name of the folder containing the model geometry that you want in the `FlightGear\data\Aircraft` folder.

## Airport ID

Specify the airport ID. The list of supported airports is available in the FlightGear interface, under **Location**.

## Runway ID

Specify the runway ID.



**Initial altitude (ft)\***

Specify the initial altitude of the aircraft, in feet. The block evaluates the value as a MATLAB expression.

**Initial heading (deg)\***

Specify the initial heading of the aircraft, in degrees. The block evaluates the value as a MATLAB expression.

**Offset distance (miles)\***

Specify the offset distance of the aircraft from the airport, in miles. The block evaluates the value as a MATLAB expression.

**Offset azimuth (deg)\***

Specify the offset azimuth of the aircraft, in degrees. The block evaluates the value as a MATLAB expression.

**Destination/Origin IP address**

Specify the network IP address of the machine on which the FlightGear software runs.

**Destination port**

Specify your network flight dynamics model (fdm) port. For more information, see the Send net\_fdm Packet to FlightGear block reference.

**Origin port**

Specify your network control (ctrl) port. For more information, see the Receive net\_ctrl Packet from FlightGear block.

**Network IP address**

Specify the network IP address of the machine on which the MATLAB software runs.

**Output file name**

Specify the name of the output file. The file name is the name of the command that you use to start FlightGear with these initial parameters. Use these file extensions:

# Generate Run Script

---

Platform	Extension
Windows	.bat
Linux and Mac OS	.sh

## FlightGear base directory

Specify the name of your FlightGear installation folder.

## Generate Script

Enter the correct information in the dialog box fields, then click **Generate Script** to generate a run script for FlightGear. Do not click this button until you have entered the correct information in the dialog box fields.

## Examples

See the asbh120 example.

## See Also

FlightGear Preconfigured 6DoF Animation

Pack net\_fdm Packet for FlightGear

Send net\_fdm Packet to FlightGear

Receive net\_ctrl Packet from FlightGear

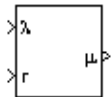
Unpack net\_ctrl Packet from FlightGear

# Geocentric to Geodetic Latitude

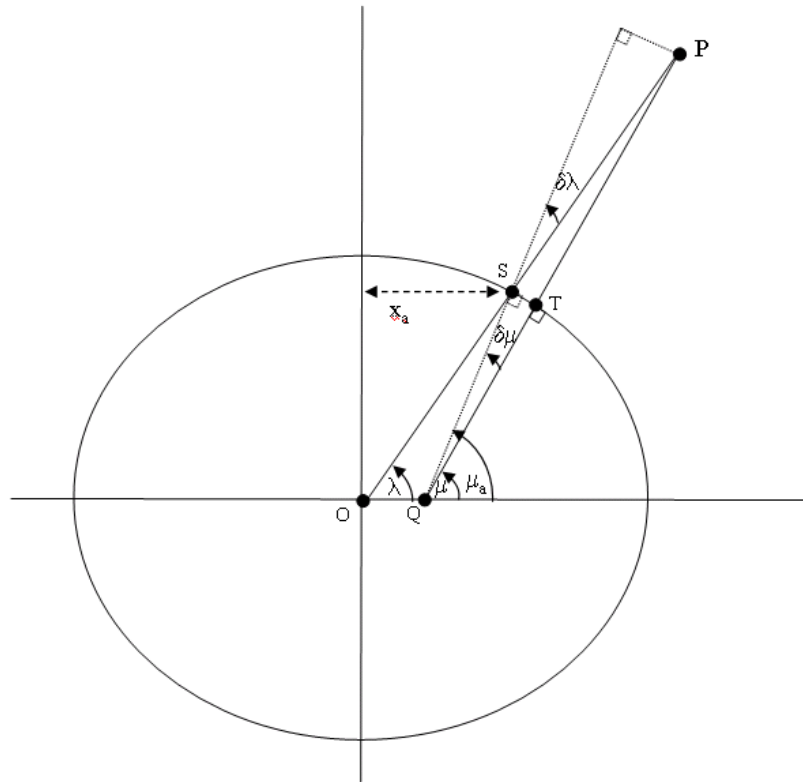
**Purpose** Convert geocentric latitude to geodetic latitude

**Library** Utilities/Axes Transformations

**Description**



The Geocentric to Geodetic Latitude block converts a geocentric latitude ( $\lambda$ ) into geodetic latitude ( $\mu$ ). There are a number of geometric relationships that are used to calculate the geodetic latitude in this noniterative method. A number of angles and points are involved in the calculation, which are shown in following figure.



# Geocentric to Geodetic Latitude

---

Given geocentric latitude ( $\lambda$ ) and the radius ( $r$ ) from the center of the planet (O) to the center of gravity (P), this noniterative method starts by computing values for the point of  $r$  that intercepts the surface of the planet (S). By rearranging the equation for an ellipse, the horizontal coordinate, ( $x_a$ ) is determined. When equatorial radius ( $R$ ), polar radius  $((1-f)R)$  and  $x_a \tan \lambda$ , are substituted for semi-major axis, semi-minor axis and vertical coordinate ( $y_a$ ), the resulting equation for  $x_a$  has the following form:

$$x_a = \frac{(1-f)R}{\sqrt{\tan^2 \lambda + (1-f)^2}}$$

To determine the geodetic latitude at S  $\mu_a$ , the equation for an ellipse with equatorial radius ( $R$ ), polar radius  $((1-f)R)$  is used again. This time it is used to define  $y_a$  in terms of  $x_a$ .

$$y_a = \sqrt{R^2 - x_a^2}(1-f)$$

Additionally, the relationship between geocentric latitude at the planet's surface and geodetic latitude is used.

$$\mu_a = \text{atan} \left( \frac{\tan \lambda}{(1-f)^2} \right)$$

Using the relationship  $\tan \lambda = y_a / x_a$  and the two equations above, the resulting equation for  $\mu_a$  is obtained.

$$\mu_a = \text{atan} \left( \frac{\sqrt{R^2 - x_a^2}}{(1-f)x_a} \right)$$

The correct sign of  $\mu_a$  is determined by testing  $\lambda$  and if  $\lambda$  is less than zero  $\mu_a$  changes sign accordingly.

In order to calculate the geodetic latitude of P, a number of geometric relationships are required to be calculated. These calculations follow.

The radius ( $r_a$ ) from the center of the planet (O) to the surface of the planet (S) is calculated by using trigonometric relationship.

$$r_a = \frac{x_a}{\cos \lambda}$$

The distance from S to P is defined by:

$$l = r - r_a$$

The angular difference between geocentric latitude and geodetic latitude at S ( $\delta\lambda$ ) is defined by:

$$\delta\lambda = \mu_a - \lambda$$

Using  $l$  and  $\delta\lambda$ , the segment TP or the mean sea-level altitude ( $h$ ) is estimated.

$$h = l \cos \delta\lambda$$

The equation for the radius of curvature in the Meridian ( $\rho_a$ ) at  $\mu_a$  is

$$\rho_a = \frac{R(1-f)^2}{\left(1 - (2f - f^2)\sin^2 \mu_a\right)^{3/2}}$$

Using  $l$ ,  $\delta\lambda$ ,  $h$ , and  $\rho_a$ , the angular difference between geodetic latitude at S ( $\mu$ ) and geodetic latitude at P ( $\mu_a$ ) is defined as:

# Geocentric to Geodetic Latitude

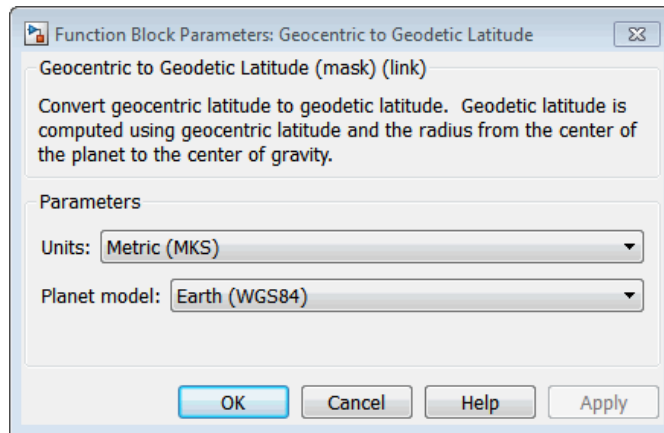
---

$$\delta\mu = \text{atan}\left(\frac{l \sin \delta\lambda}{\rho_a + h}\right)$$

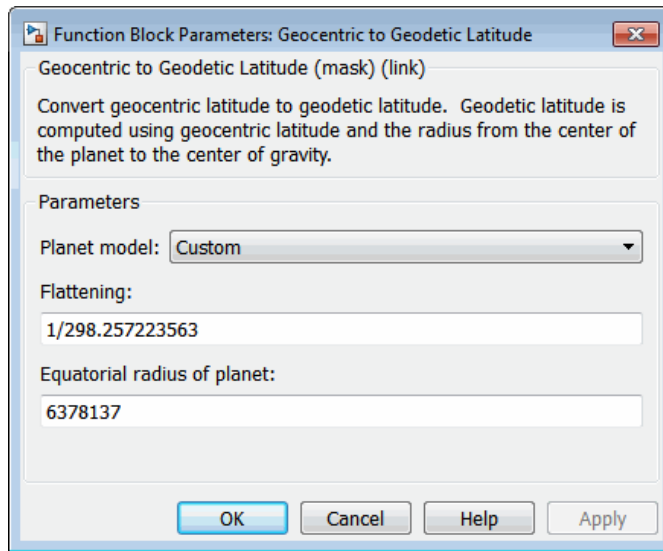
Subtracting  $\delta\mu$  from  $\mu_a$  then gives  $\mu$ .

$$\mu = \mu_a - \delta\mu$$

## Dialog Box



# Geocentric to Geodetic Latitude



## Units

Specifies the parameter and output units:

<b>Units</b>	<b>Radius from CG to Center of Planet</b>	<b>Equatorial Radius</b>
Metric (MKS)	Meters	Meters
English	Feet	Feet

This option is only available when **Planet model** is set to Earth (WGS84).

## Planet model

Specifies the planet model to use: Custom or Earth (WGS84).

## Flattening

Specifies the flattening of the planet. This option is only available with **Planet model** set to Custom.

# Geocentric to Geodetic Latitude

---

## Equatorial radius of planet

Specifies the radius of the planet at its equator. The units of the equatorial radius parameter should be the same as the units for radius. This option is only available with **Planet model** set to Custom.

## Inputs and Outputs

Input	Dimension Type	Description
First	Scalar	Contains the geocentric latitude, in degrees.
Second	Scalar	Contains the radius from center of the planet to the center of gravity.

Output	Dimension Type	Description
First	Scalar	Contains the geodetic latitude, in degrees.

## Assumptions and Limitations

This implementation generates a geodetic latitude that lies between  $\pm 90$  degrees.

## References

Jackson, E. B., *Manual for a Workstation-based Generic Flight Simulation Program (LaRCsim) Version 1.4*, NASA TM 110164, April, 1995.

Hedgley, D. R., Jr., "An Exact Transformation from Geocentric to Geodetic Coordinates for Nonzero Altitudes," NASA TR R-458, March, 1976.

Clynch, J. R., "Radius of the Earth - Radii Used in Geodesy," Naval Postgraduate School, 2002, <http://www.oc.nps.navy.mil/oc2902w/geodesy/radiigeo.pdf>.

Stevens, B. L., and F. L. Lewis, *Aircraft Control and Simulation*, John Wiley & Sons, New York, 1992.



Edwards, C. H., and D. E. Penny, *Calculus and Analytical Geometry 2nd Edition*, Prentice-Hall, Englewood Cliffs, New Jersey, 1986.

## See Also

ECEF Position to LLA

Flat Earth to LLA

Geodetic to Geocentric Latitude

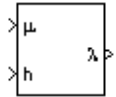
LLA to ECEF Position

# Geodetic to Geocentric Latitude

**Purpose** Convert geodetic latitude to geocentric latitude

**Library** Utilities/Axes Transformations

## Description



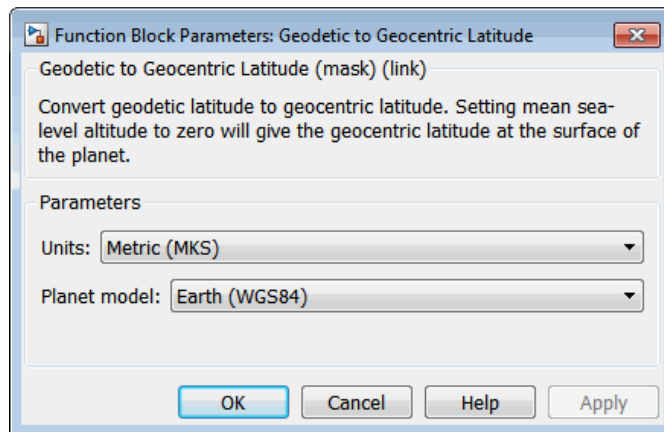
The Geodetic to Geocentric Latitude block converts a geodetic latitude ( $\mu$ ) into geocentric latitude ( $\lambda$ ). Geocentric latitude at the planet surface ( $\lambda_s$ ) is defined by flattening ( $f$ ), and geodetic latitude in the following relationship.

$$\lambda_s = \text{atan}((1 - f)^2 \tan \mu)$$

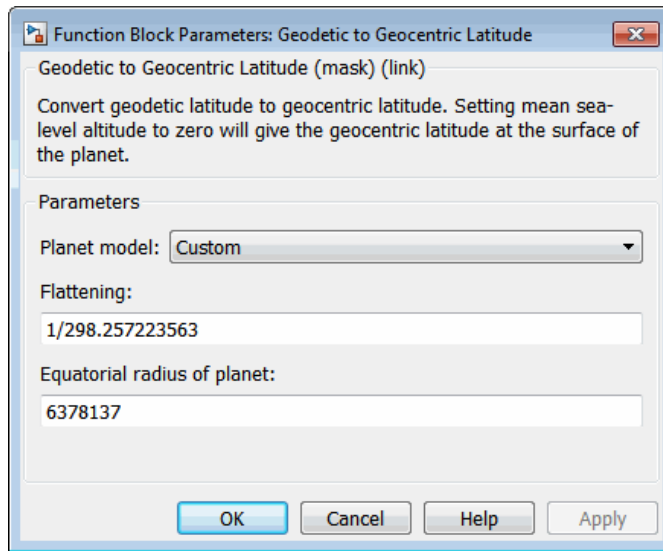
Geocentric latitude is defined by mean sea-level altitude ( $h$ ), geodetic latitude, radius of the planet ( $r_s$ ) and geocentric latitude at the planet surface in the following relationship.

$$\lambda = \text{atan} \left( \frac{h \sin \mu + r_s \sin \lambda_s}{h \cos \mu + r_s \cos \lambda_s} \right)$$

## Dialog Box



# Geodetic to Geocentric Latitude



## Units

Specifies the parameter and output units:

<b>Units</b>	<b>Altitude</b>	<b>Equatorial Radius</b>
Metric (MKS)	Meters	Meters
English	Feet	Feet

This option is only available when **Planet model** is set to Earth (WGS84).

## Planet model

Specifies the planet model to use: Custom or Earth (WGS84).

## Flattening

Specifies the flattening of the planet. This option is only available with **Planet model** set to Custom.

# Geodetic to Geocentric Latitude

---

## Equatorial radius of planet

Specifies the radius of the planet at its equator. The units of the equatorial radius parameter should be the same as the units for altitude. This option is only available with **Planet model** set to Custom.

## Inputs and Outputs

Input	Dimension Type	Description
First	Scalar	Contains the geocentric latitude, in degrees.
Second	Scalar	Contains the mean sea-level altitude (MSL).

Output	Dimension Type	Description
First	Scalar	Contains the geodetic latitude, in degrees.

## Assumptions and Limitations

This implementation generates a geocentric latitude that lies between  $\pm 90$  degrees.

## Reference

Stevens, B. L., and F. L. Lewis, *Aircraft Control and Simulation*, John Wiley & Sons, New York, 1992.

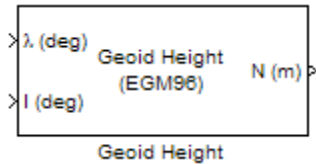
## See Also

ECEF Position to LLA  
Flat Earth to LLA  
Geocentric to Geodetic Latitude  
LLA to ECEF Position  
Radius at Geocentric Latitude

**Purpose** Calculate undulations/height

**Library** Environment/Gravity

## Description

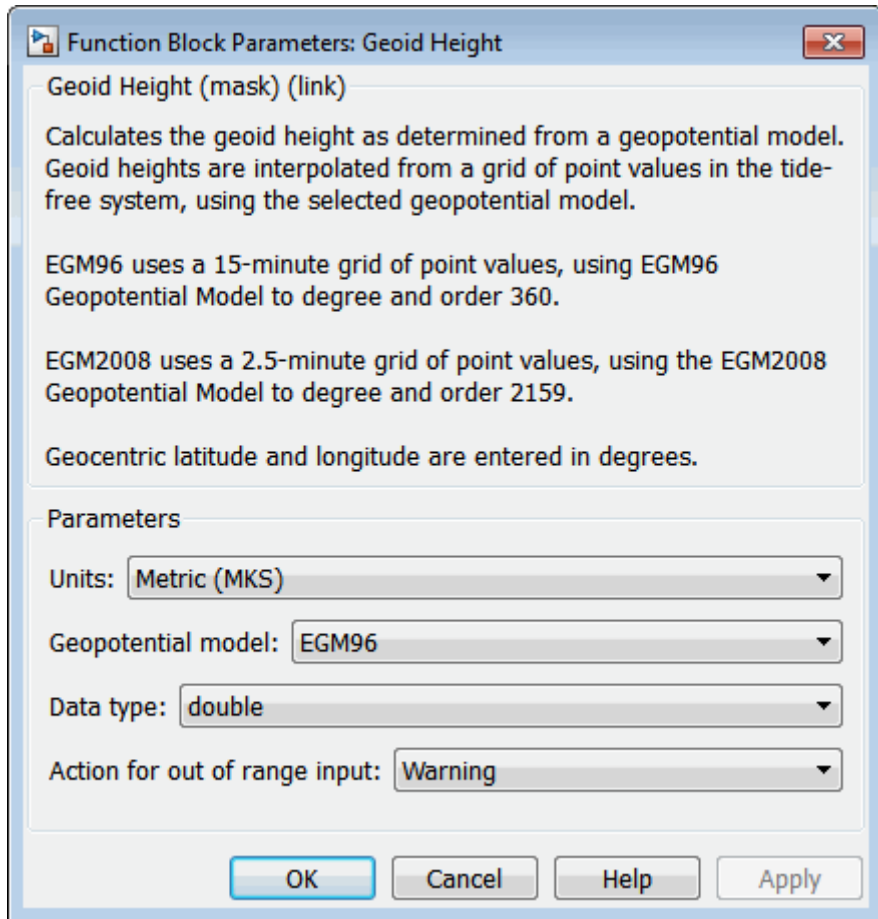


The Geoid Height block calculates the geoid height using the **Geopotential model** parameter. The block interpolates the geoid heights from a grid of point values in the tide-free system. It uses the specified geopotential model to degree and order 360. The geoid undulations are relative to the WGS84 ellipsoid.

The interpolation scheme wraps over the poles to allow for geoid height calculations at and near these locations.

# Geoid Height

## Dialog Box



### Units

Specifies the parameter and output units:

Units	Height
Metric (MKS)	Meters
English	Feet

## Geopotential model

From the list, select the geopotential model.

Geopotential Model	Description
EGM96 (Earth)	Default. EGM96 Geopotential Model to degree and order 360. This model uses a 15-minute grid of point values in the tide-free system. This block calculates geoid heights to an accuracy of 0.01 m for this model.
EGM2008 (Earth)	EGM2008 Geopotential Model to degree and order 2159. This model uses a 2.5-minute grid of point values in the tide-free system. This block calculates geoid heights to an accuracy of 0.001 m for this model.
Custom	Custom geopotential model that you define in <i>datafile</i> . This block calculates geoid heights to an accuracy of 0.01 m for custom models.

## Data type

Specifies the data type of the input and output signals. From the list, select double or single.

## Action

String that defines action for out-of-range input. Specify one:

- 'Error'
- 'Warning' (default)
- 'None'

# Geoid Height

---

## Inputs and Outputs

Input	Dimension Type	Description
First	Scalar	Contains geocentric latitude in degrees, where north latitude is positive, and south latitude is negative. Input latitude must be of type single or double. If latitude is not in the range from -90 to 90, the block wraps it to be within the range.
Second	Scalar	Contains geocentric longitude in degrees, where east longitude is positive in the range from 0 to 360. Input longitude must be of type single or double. If longitude is not in the range from 0 to 360, the block wraps it to be within the range.

Output	Dimension Type	Description
N	Scalar	Contains the geoid height in selected length units. The data type is the same as the latitude in the first input.

## Limitations

This block has the limitations of the selected geopotential model.

## References

Vallado, D. A. "Fundamentals of Astrodynamics and Applications." McGraw-Hill, New York, 1997.

NIMA TR8350.2: "Department of Defense World Geodetic System 1984, Its Definition and Relationship with Local Geodetic Systems."

National Geospatial-Intelligence Agency Web site:  
<http://earth-info.nga.mil/GandG/publications/vertdatum.html>

## See Also

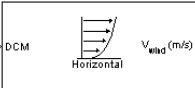
WGS84 Gravity Model, Spherical Harmonic Gravity Model



**Purpose** Transform horizontal wind into body-axes coordinates

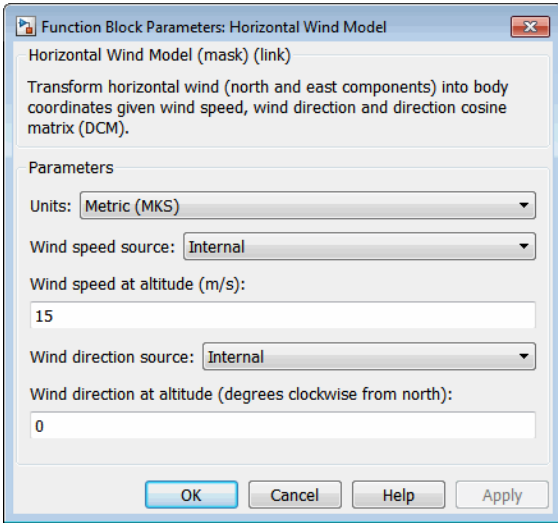
**Library** Environment/Wind

**Description** The Horizontal Wind Model block computes the wind velocity in body-axes coordinates.



The wind is specified by wind speed and wind direction in Earth axes. The speed and direction can be constant or variable over time. The direction of the wind is in degrees clockwise from the direction of the Earth  $x$ -axis (north). The wind direction is defined as the direction from which the wind is coming. Using the direction cosine matrix (DCM), the wind velocities are transformed into body-axes coordinates.

## Dialog Box



**Units** Specifies the input and output units:

# Horizontal Wind Model

---

<b>Units</b>	<b>Wind Speed</b>	<b>Wind Velocity</b>
Metric (MKS)	Meters per second	Meters per second
English (Velocity in ft/s)	Feet per second	Feet per second
English (Velocity in kts)	Knots	Knots

## **Wind speed source**

Specify source of wind speed:

External	Variable wind speed input to block
Internal	Constant wind speed specified in mask

## **Wind speed at altitude (m/s)**

Constant wind speed used if internal wind speed source is selected.

## **Wind direction source**

Specify source of wind direction:

External	Variable wind direction input to block
Internal	Constant wind direction specified in mask

## **Wind direction at altitude (degrees clockwise from north)**

Constant wind direction used if internal wind direction source is selected. The direction of the wind is in degrees clockwise from the direction of the Earth  $x$ -axis (north). The wind direction is defined as the direction from which the wind is coming.

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the direction cosine matrix.
Second (Optional)		Contains the wind speed in selected units.
Third (Optional)		Contains the wind direction in degrees.

Output	Dimension Type	Description
First		Contains the wind velocity in body-axes, in selected units.

## See Also

Dryden Wind Turbulence Model (Continuous)

Dryden Wind Turbulence Model (Discrete)

Discrete Wind Gust Model

Von Karman Wind Turbulence Model (Continuous)

Wind Shear Model

# Ideal Airspeed Correction

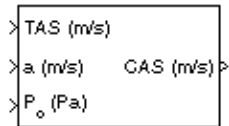
## Purpose

Calculate equivalent airspeed (EAS), calibrated airspeed (CAS), or true airspeed (TAS) from each other

## Library

Flight Parameters

## Description



The Ideal Airspeed Correction block calculates one of the following airspeeds: equivalent airspeed (EAS), calibrated airspeed (CAS), or true airspeed (TAS), from one of the other two airspeeds.

Three equations are used to implement the Ideal Airspeed Correction block. The first equation shows TAS as a function of EAS, relative pressure ratio at altitude ( $\delta$ ), and speed of sound at altitude ( $a$ ).

$$TAS = \frac{EAS \times a}{a_0 \sqrt{\delta}}$$

Using the compressible form of Bernoulli's equation and assuming isentropic conditions, the last two equations for EAS and CAS are derived.

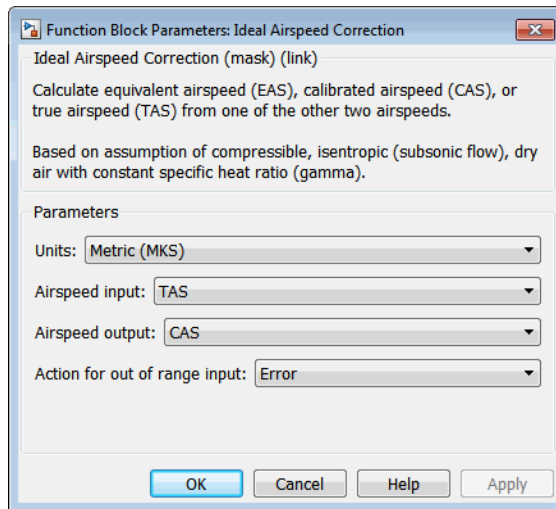
$$EAS = \sqrt{\frac{2\gamma P}{(\gamma - 1)\rho_0} \left( \left( \frac{q}{P} + 1 \right)^{(\gamma-1)/\gamma} - 1 \right)}$$

$$CAS = \sqrt{\frac{2\gamma P_0}{(\gamma - 1)\rho_0} \left( \left( \frac{q}{P_0} + 1 \right)^{(\gamma-1)/\gamma} - 1 \right)}$$

In order to generate a correction table and its approximate inverse, these two equations were solved for dynamic pressure ( $q$ ). Having values of  $q$  by a function of  $EAS$  and ambient pressure at altitude ( $P$ ) or by a function of  $CAS$ , allows the two equations to be solved using the other's solution for  $q$ , thus creating a solution for  $EAS$  that depends on  $P$  and  $CAS$  and a solution for  $CAS$  that depends on  $P$  and  $EAS$ .

# Ideal Airspeed Correction

## Dialog Box



### Units

Specifies the input and output units:

Units	Airspeed Input	Speed of Sound	Air Pressure	Airspeed Output
Metric (MKS)	Meters per second	Meters per second	Pascal	Meters per second
English (Velocity in ft/s)	Feet per second	Feet per second	Pound force per square inch	Feet per second
English (Velocity in kts)	Knots	Knots	Pound force per square inch	Knots

# Ideal Airspeed Correction

---

## Airspeed input

Specify the airspeed input type:

TAS	True airspeed
EAS	Equivalent airspeed
CAS	Calibrated airspeed

## Airspeed output

Specify the airspeed output type:

<b>Velocity Input</b>	<b>Velocity Output</b>
TAS	EAS (equivalent airspeed) CAS (calibrated airspeed)
EAS	TAS (true airspeed) CAS (calibrated airspeed)
CAS	TAS (true airspeed) EAS (equivalent airspeed)

## Action for out of range input

Specify if an out-of-range input (supersonic airspeeds) invokes a warning, an error, or no action.

## Inputs and Outputs

<b>Input</b>	<b>Dimension Type</b>	<b>Description</b>
First		Contains the selected airspeed in the selected units.
Second		Contains the speed of sound in the selected units.
Third		Contains the static pressure in the selected units.

Output	Dimension Type	Description
First		Contains the selected airspeed in the selected units.

## Assumptions and Limitations

This block assumes that the air flow is compressible, isentropic (subsonic flow), dry air with constant specific heat ratio,  $\gamma$ .

## Examples

See the `aeroblk_indicated` model and the `aeroblk_calibrated` model for examples of this block.

## References

Lowry, J. T., *Performance of Light Aircraft*, AIAA Education Series, Washington, DC, 1999.

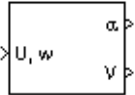
*Aeronautical Vestpocket Handbook*, United Technologies Pratt & Whitney, August, 1986.

# Incidence & Airspeed

**Purpose** Calculate incidence and airspeed

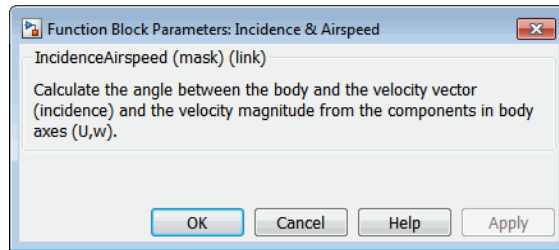
**Library** Flight Parameters

**Description** The Incidence & Airspeed block supports the 3DoF equations of motion model by calculating the angle between the velocity vector and the body, and also the total airspeed from the velocity components in the body-fixed coordinate frame.



$$\alpha = \text{atan}\left(\frac{w}{u}\right)$$
$$V = \sqrt{u^2 + w^2}$$

## Dialog Box



## Inputs and Outputs

Input	Dimension Type	Description
First	Two-element vector	Contains the velocity of the body resolved into the body-fixed coordinate frame.

Output	Dimension Type	Description
First		Contains the incidence angle, in radians.
Second		Contains the airspeed of the body.



**Examples**

See the Aerodynamics & Equations of Motion subsystem of the `aeroblk_guidance_airframe` model for examples of this block.

**See Also**

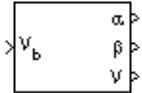
Incidence, Sideslip & Airspeed

# Incidence, Sideslip & Airspeed

**Purpose** Calculate incidence, sideslip, and airspeed

**Library** Flight Parameters

**Description** The Incidence, Sideslip & Airspeed block supports the 6DoF (Euler Angles) and 6DoF (Quaternion) models by calculating the angles between the velocity vector and the body, and also the total airspeed from the velocity components in the body-fixed coordinate frame.



$$\alpha = a \tan\left(\frac{w}{u}\right)$$

$$\beta = a \sin\left(\frac{v}{V}\right)$$

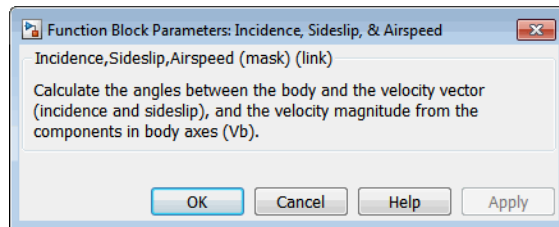
$$V = \sqrt{u^2 + v^2 + w^2}$$

$$\alpha = \text{atan}\left(\frac{w}{u}\right)$$

$$\beta = \text{asin}\left(\frac{v}{V}\right)$$

$$V = \sqrt{u^2 + v^2 + w^2}$$

## Dialog Box



# Incidence, Sideslip & Airspeed

---

## Inputs and Outputs

Input	Dimension Type	Description
First	Three-element vector	Contains the velocity of the body resolved into the body-fixed coordinate frame.

Output	Dimension Type	Description
First		Contains the incidence angle in radians.
Second		Contains the sideslip angle in radians.
Third		Contains the airspeed of the body.

## Examples

See Airframe in the aeroblk\_HL20 model for an example of this block.

## See Also

Incidence & Airspeed

# International Geomagnetic Reference Field 11

---

**Purpose** Calculate Earth's magnetic field and secular variation using 11th generation of International Geomagnetic Reference Field

**Library** Environment/Gravity

## Description

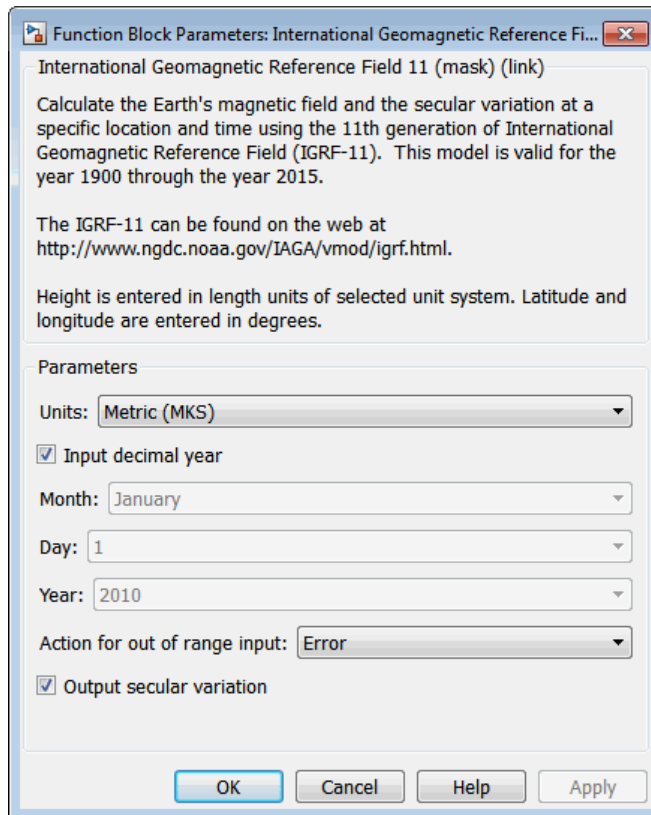
```
> h (m)           Magnetic Field (nT) >
                  Horizontal Intensity (nT) >
                  Declination (deg) >
> μ (deg)         Inclination (deg) >
                  Total Intensity (nT) >
                  SV Magnetic Field (nT/yr) >
> I (deg)         SV Horizontal Intensity (nT/yr) >
                  SV Declination (min/yr) >
                  SV Inclination (min/yr) >
> Decimal Year    SV Total Intensity (nT/yr) >
```

International Geomagnetic Reference Field 11

The International Geomagnetic Reference Field 11 block calculates the Earth's magnetic field and secular variation using the 11th generation of the International Geomagnetic Reference Field. It calculates these values at a location and time that you define.

# International Geomagnetic Reference Field 11

## Dialog Box



## Units

Specifies the parameter and output units:

Units	Height
Metric (MKS)	Meters
English	Feet

# International Geomagnetic Reference Field 11

---

## Input decimal year

When selected, the decimal year is an input for the International Geomagnetic Reference Field 11 block. Otherwise, specify a date using the dialog parameters **Month**, **Day**, and **Year**.

## Month

Specifies the month to calculate decimal year.

## Day

Specifies the day to calculate decimal year.

## Year

Specifies the year to calculate decimal year. In the list, select from 1900 to 2015.

## Action for out of range input

Specifies whether out-of-range input invokes a warning, error, or no action.

## Output secular variance

Select this check box to enable the output of secular variances (annual rate of change) with nonsecular variances:

Secular Variance	Description
Magnetic Field	Magnetic field vector, in nanotesla (nT). <i>Z</i> is the vertical component (+ve down).
Horizontal Intensity	Horizontal intensity, in nanotesla (nT).
Declination	Declination, in degrees (+ve east).
Inclination	Inclination, in degrees (+ve down).
Total Intensity	Total intensity, in nanotesla (nT).

# International Geomagnetic Reference Field 11

Secular Variance	Description
SV Magnetic Field	Secular variation of magnetic field.
SV Horizontal Intensity	Secular variation of horizontal intensity.
SV Declination	Secular variation of declination, the angle between true north and the magnetic field vector (positive eastward).
SV Inclination	Secular variation of inclination, the angle between the horizontal plane and the magnetic field vector (positive downward).
SV Total Intensity	Secular variation of total intensity.

Clear this check box to enable just the nonsecular variances:

- Magnetic Field
- Horizontal Intensity
- Declination
- Inclination
- Total Intensity

## Inputs and Outputs

Input	Dimension Type	Description
First	Scalar	Contains the height, in selected units.
Second	Scalar	Contains the latitude, in degrees.

# International Geomagnetic Reference Field 11

---

Input	Dimension Type	Description
Third	Scalar	Contains the longitude, in degrees.
Fourth (Optional)	Scalar	Contains the desired year in a decimal format to include any fraction of the year that has already passed. The value is the current year plus the number of days that have passed in this year divided by 365. The following code illustrates how to calculate the decimal year, dyear, for March 21, 2005:  %%BEGIN CODE%% dyear=decyear('21-March-2005','dd-mmm-yyyy') %%END CODE%%

Output	Dimension Type	Description
First		Contains the magnetic field vector, in selected units.
Second		Contains the horizontal intensity, in selected units.
Third		Contains the declination, in degrees.
Fourth		Contains the inclination, in degrees.
Fifth		Contains the total intensity, in selected units.
Sixth (Optional)		Contains the secular variation of magnetic field vector, in selected units per years.
Seventh (Optional)		Contains the secular variation of horizontal intensity, in selected units per year.



# International Geomagnetic Reference Field 11

Output	Dimension Type	Description
Eight (Optional)		Contains the secular variation of declination, in minutes per year.
Ninth (Optional)		Contains the secular variation of inclination, in minutes per year.
Tenth (Optional)		Contains the secular variation of total intensity, in selected units per year.

## Limitations

This block is valid between the heights of -1000 m and 600000 m.

This block is valid between the years of 1900 and 2015.

The following site has additional limitations:

<http://www.ngdc.noaa.gov/IAGA/vmod/igrfhw.html>

## References

International Association of Geomagnetism and Aeronomy.  
11th Generation International Geomagnetic Reference Field:  
<http://www.ngdc.noaa.gov/IAGA/vmod/igrf.html>

Blakely, R. J., *Potential Theory in Gravity & Magnetic Applications*.  
Cambridge, UK: Cambridge University Press, 1996.

# Interpolate Matrix(x)

---

**Purpose** Return interpolated matrix for given input

**Library** GNC/Controls

**Description** The Interpolate Matrix(x) block interpolates a one-dimensional array of matrices.



This one-dimensional case assumes a matrix  $M$  is defined at a discrete number of values of an independent variable

$$x = [x_1 \ x_2 \ x_3 \ \dots \ x_i \ x_{i+1} \ \dots \ x_n].$$

Then for  $x_i < x < x_{i+1}$ , the block output is given by

$$(1 - \lambda)M(x_i) + \lambda M(x_{i+1})$$

where the interpolation fraction is defined as

$$\lambda = (x - x_i) / (x_{i+1} - x_i)$$

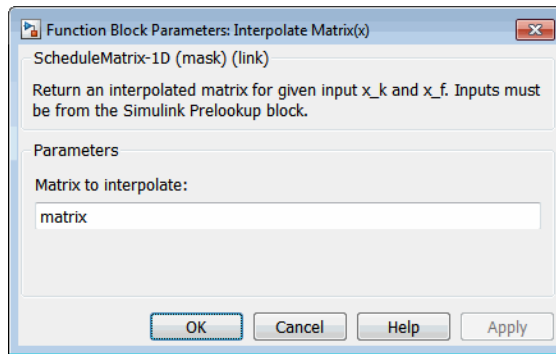
The matrix to be interpolated should be three dimensional, the first two dimensions corresponding to the matrix at each value of  $x$ . For example, if you have three matrices  $A$ ,  $B$ , and  $C$  defined at  $x = 0$ ,  $x = 0.5$ , and  $x = 1.0$ , then the input matrix is given by

```
matrix(:, :, 1) = A;
```

```
matrix(:, :, 2) = B;
```

```
matrix(:, :, 3) = C;
```

## Dialog Box



## Matrix to interpolate

Matrix to be interpolated, with three indices and the third index labeling the interpolating values of  $x$ .

## Inputs and Outputs

Input	Dimension	Type	Description
First			Contains the interpolation index $i$ .
Second			Contains the interpolation fraction $\lambda$ .

Output	Dimension	Type	Description
First			Contains the interpolated matrix.

## Assumptions and Limitations

This block must be driven from the Simulink Prelookup block.

## Examples

See the following block reference pages: 1D Controller  $[A(v), B(v), C(v), D(v)]$ , 1D Observer Form  $[A(v), B(v), C(v), F(v), H(v)]$ , and 1D Self-Conditioned  $[A(v), B(v), C(v), D(v)]$ .

## See Also

Interpolate Matrix(x,y)

# Interpolate Matrix(**x**)

---

Interpolate Matrix(x,y,z)

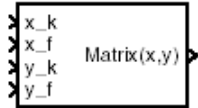
## Purpose

Return interpolated matrix for given inputs

## Library

GNC/Controls

## Description



The Interpolate Matrix(x,y) block interpolates a two-dimensional array of matrices.

This two-dimensional case assumes the matrix is defined as a function of two independent variables,  $\mathbf{x} = [x_1 \ x_2 \ x_3 \dots \ x_i \ x_{i+1} \dots \ x_n]$  and  $\mathbf{y} = [y_1 \ y_2 \ y_3 \dots \ y_j \ y_{j+1} \dots \ y_m]$ . For given values of  $x$  and  $y$ , four matrices are interpolated. Then for  $x_i < x < x_{i+1}$  and  $y_j < y < y_{j+1}$ , the output matrix is given by

$$(1 - \lambda_y)[(1 - \lambda_x)M(x_i, y_j) + \lambda_x M(x_{i+1}, y_j)] + \lambda_y[(1 - \lambda_x)M(x_i, y_{j+1}) + \lambda_x M(x_{i+1}, y_{j+1})]$$

where the two interpolation fractions are denoted by

$$\lambda_x = (x - x_i) / (x_{i+1} - x_i)$$

and

$$\lambda_y = (y - y_j) / (y_{j+1} - y_j)$$

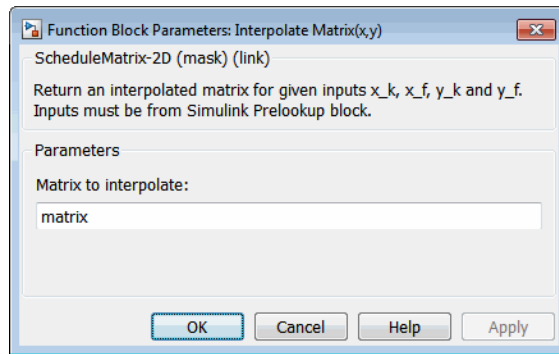
In the two-dimensional case, the interpolation is carried out first on  $x$  and then  $y$ .

The matrix to be interpolated should be four dimensional, the first two dimensions corresponding to the matrix at each value of  $x$  and  $y$ . For example, if you have four matrices  $A$ ,  $B$ ,  $C$ , and  $D$  defined at  $(x = 0.0, y = 1.0)$ ,  $(x = 0.0, y = 3.0)$ ,  $(x = 1.0, y = 1.0)$  and  $(x = 1.0, y = 3.0)$ , then the input matrix is given by

```
matrix(:,:,1,1) = A;
matrix(:,:,1,2) = B;
matrix(:,:,2,1) = C;
matrix(:,:,2,2) = D;
```

# Interpolate Matrix(x,y)

## Dialog Box



## Matrix to interpolate

Matrix to be interpolated, with four indices and the third and fourth indices labeling the interpolating values of  $x$  and  $y$ .

## Inputs and Outputs

Input	Dimension	Type	Description
First			Contains the first interpolation index $i$ .
Second			Contains the first interpolation fraction $\lambda_x$ .
Third			Contains the second interpolation index $j$ .
Fourth			Contains the second interpolation fraction $\lambda_y$ .

Output	Dimension	Type	Description
First			Contains the interpolated matrix.

## Assumptions and Limitations

This block must be driven from the Simulink Prelookup block.

## Examples

See the following block reference pages: 2D Controller [A(v),B(v),C(v),D(v)], 2D Observer Form [A(v),B(v),C(v),F(v),H(v)], and 2D Self-Conditioned [A(v),B(v),C(v),D(v)].

## See Also

Interpolate Matrix(x)

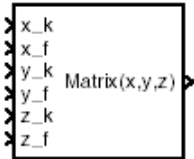
Interpolate Matrix(x,y,z)

# Interpolate Matrix(x,y,z)

**Purpose** Return interpolated matrix for given inputs

**Library** GNC/Controls

**Description** The Interpolate Matrix(x,y,z) block interpolates a three-dimensional array of matrices.



This three-dimensional case assumes the matrix is defined as a function of three independent variables:

$$x = [x_1 \ x_2 \ x_3 \ \dots \ x_i \ x_{i+1} \ \dots \ x_n]$$

$$y = [y_1 \ y_2 \ y_3 \ \dots \ y_j \ y_{j+1} \ \dots \ y_m]$$

$$z = [z_1 \ z_2 \ z_3 \ \dots \ z_k \ z_{k+1} \ \dots \ z_p]$$

For given values of  $x$ ,  $y$ , and  $z$ , eight matrices are interpolated. Then for

$$x_i < x < x_{i+1}$$

$$y_j < y < y_{j+1}$$

$$z_k < z < z_{k+1}$$

the output matrix is given by

$$\begin{aligned} & (1 - \lambda_z) \left\{ (1 - \lambda_y) \left[ (1 - \lambda_x) M(x_i, y_j, z_k) + \lambda_x M(x_{i+1}, y_j, z_k) \right] \right. \\ & \quad \left. + \lambda_y \left[ (1 - \lambda_x) M(x_i, y_{j+1}, z_k) + \lambda_x M(x_{i+1}, y_{j+1}, z_k) \right] \right\} \\ & + \lambda_z \left\{ (1 - \lambda_y) \left[ (1 - \lambda_x) M(x_i, y_j, z_{k+1}) + \lambda_x M(x_{i+1}, y_j, z_{k+1}) \right] \right. \\ & \quad \left. + \lambda_y \left[ (1 - \lambda_x) M(x_i, y_{j+1}, z_{k+1}) + \lambda_x M(x_{i+1}, y_{j+1}, z_{k+1}) \right] \right\} \end{aligned}$$

where the three interpolation fractions are denoted by



$$\lambda_x = (x - x_i) / (x_{i+1} - x_i)$$

$$i_y = (y - y_j) / (y_{j+1} - y_j)$$

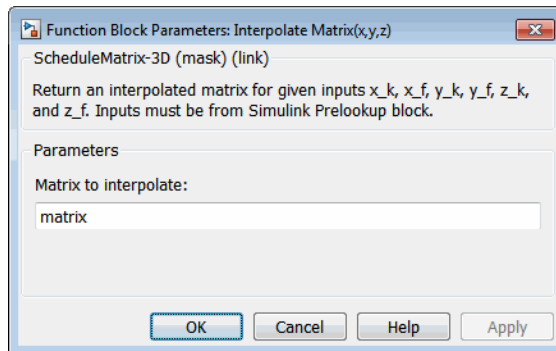
$$\lambda_z = (z - z_k) / (z_{k+1} - z_k)$$

In the three-dimensional case, the interpolation is carried out first on  $x$ , then  $y$ , and finally  $z$ .

The matrix to be interpolated should be five dimensional, the first two dimensions corresponding to the matrix at each value of  $x$ ,  $y$ , and  $z$ . For example, if you have eight matrices  $A$ ,  $B$ ,  $C$ ,  $D$ ,  $E$ ,  $F$ ,  $G$ , and  $H$  defined at the following values of  $x$ ,  $y$ , and  $z$ , then the corresponding input matrix is given by

(x = 0.0, y = 1.0, z = 0.1)	matrix(:, :, 1, 1, 1) = A;
(x = 0.0, y = 1.0, z = 0.5)	matrix(:, :, 1, 1, 2) = B;
(x = 0.0, y = 3.0, z = 0.1)	matrix(:, :, 1, 2, 1) = C;
(x = 0.0, y = 3.0, z = 0.5)	matrix(:, :, 1, 2, 2) = D;
(x = 1.0, y = 1.0, z = 0.1)	matrix(:, :, 2, 1, 1) = E;
(x = 1.0, y = 1.0, z = 0.5)	matrix(:, :, 2, 1, 2) = F;
(x = 1.0, y = 3.0, z = 0.1)	matrix(:, :, 2, 2, 1) = G;
(x = 1.0, y = 3.0, z = 0.5)	matrix(:, :, 2, 2, 2) = H;

## Dialog Box



# Interpolate Matrix(**x,y,z**)

## Matrix to interpolate

Matrix to be interpolated, with five indices and the third, fourth, and fifth indices labeling the interpolating values of  $x$ ,  $y$ , and  $z$ .

## Inputs and Outputs

Input	Dimension	Type	Description
First			Contains the first interpolation index $i$ .
Second			Contains the first interpolation fraction $\lambda_x$ .
Third			Contains the second interpolation index $j$ .
Fourth			Contains the second interpolation fraction $\lambda_y$ .
Fifth			Contains the third interpolation index $k$ .
Sixth			Contains the third interpolation fraction $\lambda_z$ .

Output	Dimension	Type	Description
First			Contains the interpolated matrix.

## Assumptions and Limitations

This block must be driven from the Simulink Prelookup block.

## Examples

See the following block reference pages: 3D Controller [A(v),B(v),C(v),D(v)], 3D Observer Form [A(v),B(v),C(v),F(v),H(v)], and 3D Self-Conditioned [A(v),B(v),C(v),D(v)].

**See Also**

Interpolate Matrix(x)

Interpolate Matrix(x,y)

# Invert 3x3 Matrix

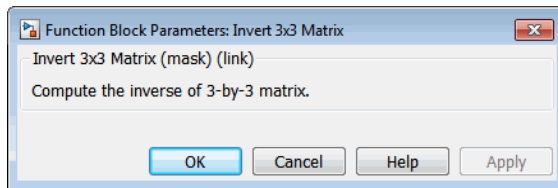
**Purpose** Compute inverse of 3-by-3 matrix

**Library** Utilities/Math Operations

**Description** The Invert 3x3 Matrix block computes the inverse of 3-by-3 matrix. If  $\det(A) = 0$ , an error occurs and the simulation stops.



## Dialog Box



## Inputs and Outputs

Input	Dimension Type	Description
First	3-by-3 matrix	

Output	Dimension Type	Description
First	3-by-3 matrix	Contains the matrix inverse of input matrix.

## See Also

Adjoint of 3x3 Matrix

Create 3x3 Matrix

Determinant of 3x3 Matrix

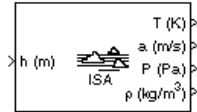
## Purpose

Implement International Standard Atmosphere (ISA)

## Library

Environment/Atmosphere

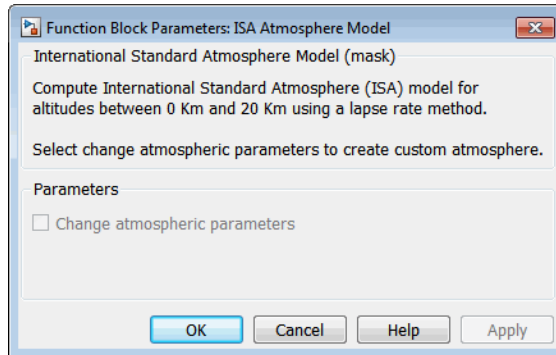
## Description



The ISA Atmosphere Model block implements the mathematical representation of the international standard atmosphere values for ambient temperature, pressure, density, and speed of sound for the input geopotential altitude.

The ISA Atmosphere Model block icon displays the input and output metric units.

## Dialog Box



## Change atmospheric parameters

Select to customize various atmospheric parameters to be different from the ISA values.

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the geopotential height.

# ISA Atmosphere Model

---

Output	Dimension Type	Description
First		Contains the temperature.
Second		Contains the speed of sound.
Third		Contains the air pressure.
Fourth		Contains the air density.

## Assumptions and Limitations

Below the geopotential altitude of 0 km and above the geopotential altitude of 20 km, temperature and pressure values are held. Density and speed of sound are calculated using a perfect gas relationship.

## Reference

[1] U.S. Standard Atmosphere, 1976, U.S. Government Printing Office, Washington, D.C.

## See Also

COESA Atmosphere Model, CIRA-86 Atmosphere Model, Lapse Rate Model

## Purpose

Transform position and velocity components from Standard Julian Epoch (J2000) to discontinued Standard Besselian Epoch (B1950)

## Library

Utilities/Axes Transformations

## Description

$$\begin{array}{|c|c|} \hline \bar{r}_{J2000} & \bar{r}_{B1950} \\ \hline \bar{v}_{J2000} & \bar{v}_{B1950} \\ \hline \end{array}$$

The Julian Epoch to Besselian Epoch block transforms two 3-by-1 vectors of Julian Epoch position ( $\bar{r}_{J2000}$ ), and Julian Epoch velocity ( $\bar{v}_{J2000}$ ) into Besselian Epoch position ( $\bar{r}_{B1950}$ ), and Besselian Epoch velocity ( $\bar{v}_{B1950}$ ). The transformation is calculated using:

$$\begin{bmatrix} \bar{r}_{B1950} \\ \bar{v}_{B1950} \end{bmatrix} = \begin{bmatrix} \bar{M}_{rr} & \bar{M}_{vr} \\ \bar{M}_{rv} & \bar{M}_{vv} \end{bmatrix}^T \begin{bmatrix} \bar{r}_{J2000} \\ \bar{v}_{J2000} \end{bmatrix}$$

where

$$(\bar{M}_{rr}, \bar{M}_{vr}, \bar{M}_{rv}, \bar{M}_{vv})$$

are defined as:

$$\bar{M}_{rr} = \begin{bmatrix} 0.9999256782 & -0.0111820611 & -0.0048579477 \\ 0.0111820610 & 0.9999374784 & -0.0000271765 \\ 0.0048579479 & -0.0000271474 & 0.9999881997 \end{bmatrix}$$

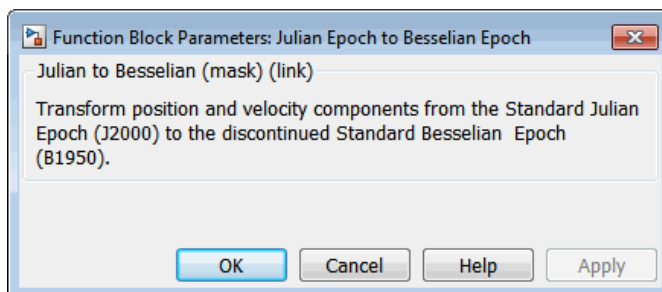
$$\bar{M}_{vr} = \begin{bmatrix} 0.00000242395018 & -0.00000002710663 & -0.00000001177656 \\ 0.00000002710663 & 0.00000242397878 & -0.00000000006587 \\ 0.00000001177656 & -0.00000000006582 & 0.00000242410173 \end{bmatrix}$$

$$\bar{M}_{rv} = \begin{bmatrix} -0.000551 & -0.238565 & 0.435739 \\ 0.238514 & -0.002667 & -0.008541 \\ -0.435623 & 0.012254 & 0.002117 \end{bmatrix}$$

# Julian Epoch to Besselian Epoch

$$\bar{M}_{vv} = \begin{bmatrix} 0.99994704 & -0.01118251 & -0.00485767 \\ 0.01118251 & 0.99995883 & -0.00002718 \\ 0.00485767 & -0.00002714 & 1.00000956 \end{bmatrix}$$

## Dialog Box



## Inputs and Outputs

Input	Dimension Type	Description
First	3-by-1 vector	Contains the position in Standard Julian Epoch (J2000).
Second	3-by-1 vector	Contains the velocity in Standard Julian Epoch (J2000).

Output	Dimension Type	Description
First	3-by-1 vector	Contains the position in Standard Besselian Epoch (B1950).
Second	3-by-1 vector	Contains the velocity in Standard Besselian Epoch (B1950).

## Reference

“Supplement to Department of Defense World Geodetic System 1984 Technical Report: Part I - Methods, Techniques and Data Used in WGS84 Development,” DMA TR8350.2-A.



# Julian Epoch to Besselian Epoch

---

## **See Also**

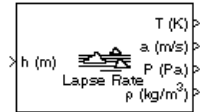
Besselian Epoch to Julian Epoch

# Lapse Rate Model

**Purpose** Implement lapse rate model for atmosphere

**Library** Environment/Atmosphere

## Description



The Lapse Rate Model block implements the mathematical representation of the lapse rate atmospheric equations for ambient temperature, pressure, density, and speed of sound for the input geopotential altitude. You can customize this atmospheric model, described below, by specifying atmospheric properties in the block dialog.

The following equations define the troposphere

$$T = T_0 - Lh$$

$$P = P_0 \left( \frac{T}{T_0} \right)^{\frac{g}{LR}}$$

$$\rho = \rho_0 \left( \frac{T}{T_0} \right)^{\frac{g}{LR}-1}$$

$$a = \sqrt{\gamma RT}$$

The following equations define the tropopause (lower stratosphere)

$$T = T_0 - Lhts$$

$$P = P_0 \left( \frac{T}{T_0} \right)^{\frac{g}{LR}} e^{\frac{g}{RT}(hts-h)}$$

$$\rho = \rho_0 \left( \frac{T}{T_0} \right)^{\frac{g}{LR}-1} e^{\frac{g}{RT}(hts-h)}$$

$$a = \sqrt{\gamma RT}$$

where:

$T_0$	Absolute temperature at mean sea level in kelvin (K)
$\rho_0$	Air density at mean sea level in kg/m <sup>3</sup>
$P_0$	Static pressure at mean sea level in N/m <sup>2</sup>
$h$	Altitude in m
$h_{ts}$	Height of the troposphere in m
$T$	Absolute temperature at altitude $h$ in kelvin (K)
$\rho$	Air density at altitude $h$ in kg/m <sup>3</sup>
$P$	Static pressure at altitude $h$ in N/m <sup>2</sup>
$a$	Speed of sound at altitude $h$ in m/s <sup>2</sup>
$L$	Lapse rate in K/m
$R$	Characteristic gas constant J/kg-K
$\gamma$	Specific heat ratio
$g$	Acceleration due to gravity in m/s <sup>2</sup>

The Lapse Rate Model block icon displays the input and output metric units.

# Lapse Rate Model

## Dialog Box

Function Block Parameters: Lapse Rate Model

International Standard Atmosphere Model (mask) (link)

Compute International Standard Atmosphere (ISA) model for altitudes between 0 Km and 20 Km using a lapse rate method.

Select change atmospheric parameters to create custom atmosphere.

Parameters

Change atmospheric parameters

Acceleration due to gravity ( $m/s^2$ ):  
9.80665

Ratio of specific heats:  
1.4

Characteristic gas constant ( $J/Kg/K$ ):  
287.0531

Lapse rate ( $K/m$ ):  
0.0065

Height of troposphere (m):  
11000

Height of tropopause (m):  
20000

Air density at mean sea level ( $Kg/m^3$ ):  
1.225

Ambient pressure at mean sea level ( $N/m^2$ ):  
101325

Ambient temperature at mean sea level (K):  
288.15

Lowest altitude (m):  
0

OK Cancel Help Apply

### Change atmospheric parameters

When selected, the following atmospheric parameters can be customized to be different from the ISA values.

**Acceleration due to gravity**

Specify the acceleration due to gravity ( $g$ ).

**Ratio of specific heats**

Specify the ratio of specific heats  $\gamma$ .

**Characteristic gas constant**

Specify the characteristic gas constant ( $R$ ).

**Lapse rate**

Specify the lapse rate of the troposphere ( $L$ ).

**Height of troposphere**

Specify the upper altitude of the troposphere, a range of decreasing temperature.

**Height of tropopause**

Specify the upper altitude of the tropopause, a range of constant temperature.

**Air density at mean sea level**

Specify the air density at sea level ( $\rho_0$ ).

**Ambient pressure at mean sea level**

Specify the ambient pressure at sea level ( $P_0$ ).

**Ambient temperature at mean sea level**

Specify the ambient temperature at sea level ( $T_0$ ).

**Lowest altitude (m)**

Specify the lowest altitude above which temperature and pressure lapse. **Lowest altitude (m)** must be below **Height of tropopause**. Default value is 0 m.

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the geopotential height.

# Lapse Rate Model

---

Output	Dimension Type	Description
First		Contains the temperature.
Second		Contains the speed of sound.
Third		Contains the air pressure.
Fourth		Contains the air density.

## Assumptions and Limitations

Below the geopotential altitude of 0 km and above the geopotential altitude of the tropopause, temperature and pressure values are held. Density and speed of sound are calculated using a perfect gas relationship.

## Reference

[1] U.S. Standard Atmosphere, 1976, U.S. Government Printing Office, Washington, D.C.

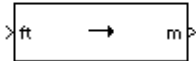
## See Also

COESA Atmosphere Model  
ISA Atmosphere Model

**Purpose** Convert from length units to desired length units

**Library** Utilities/Unit Conversions

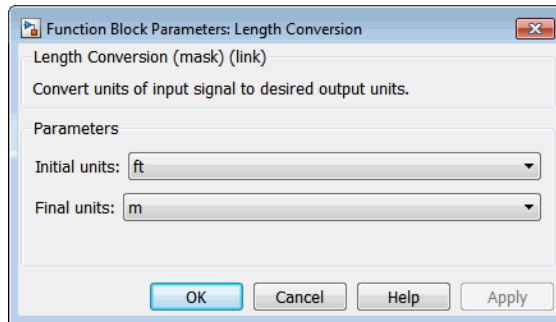
## Description



The Length Conversion block computes the conversion factor from specified input length units to specified output length units and applies the conversion factor to the input signal.

The Length Conversion block icon displays the input and output units selected from the **Initial units** and the **Final units** lists.

## Dialog Box



### Initial units

Specifies the input units.

### Final units

Specifies the output units.

The following conversion units are available:

m	Meters
ft	Feet
km	Kilometers
in	Inches

# Length Conversion

---

mi Miles  
naut mi Nautical miles

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the length in initial length units.

Output	Dimension Type	Description
First		Contains the length in final length units.

## See Also

Acceleration Conversion  
Angle Conversion  
Angular Acceleration Conversion  
Angular Velocity Conversion  
Density Conversion  
Force Conversion  
Mass Conversion  
Pressure Conversion  
Temperature Conversion  
Velocity Conversion



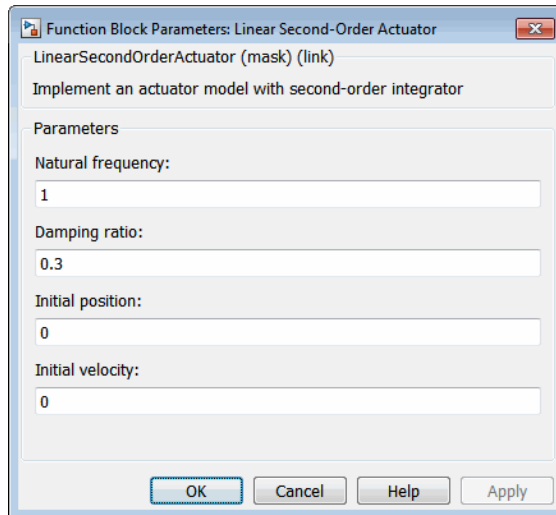
**Purpose** Implement second-order linear actuator

**Library** Actuators

**Description** The Second Order Linear Actuator block outputs the actual actuator position using the input demanded actuator position and other dialog box parameters that define the system.



## Dialog Box



### Natural frequency

The natural frequency of the actuator. The units of natural frequency are radians per second.

### Damping ratio

The damping ratio of the actuator. A dimensionless parameter.

### Initial position

The initial position of the actuator. The units of initial position must be the same as the units of demanded actuator position.

# Linear Second-Order Actuator

---

## Initial velocity

The initial velocity of the actuator. The units of initial velocity must be the same as the units of demanded actuator velocity per second.

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the demanded actuator position.

Output	Dimension Type	Description
First		Contains the actual actuator position.

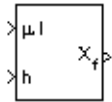
## See Also

Nonlinear Second-Order Actuator

**Purpose** Calculate Earth-centered Earth-fixed (ECEF) position from geodetic latitude, longitude, and altitude above planetary ellipsoid

**Library** Utilities/Axes Transformations

**Description** The LLA to ECEF Position block converts geodetic latitude ( $\bar{\mu}$ ),



longitude ( $\bar{\iota}$ ), and altitude ( $\bar{h}$ ) above the planetary ellipsoid into a 3-by-1 vector of ECEF position ( $\bar{p}$ ). The ECEF position is calculated from geocentric latitude at mean sea-level ( $\lambda_s$ ) and longitude using:

$$\bar{p} = \begin{bmatrix} \bar{p}_x \\ \bar{p}_y \\ \bar{p}_z \end{bmatrix} = \begin{bmatrix} r_s \cos \lambda_s \cos \iota + h \cos \mu \cos \iota \\ r_s \cos \lambda_s \sin \iota + h \cos \mu \sin \iota \\ r_s \sin \lambda_s + h \sin \mu \end{bmatrix}$$

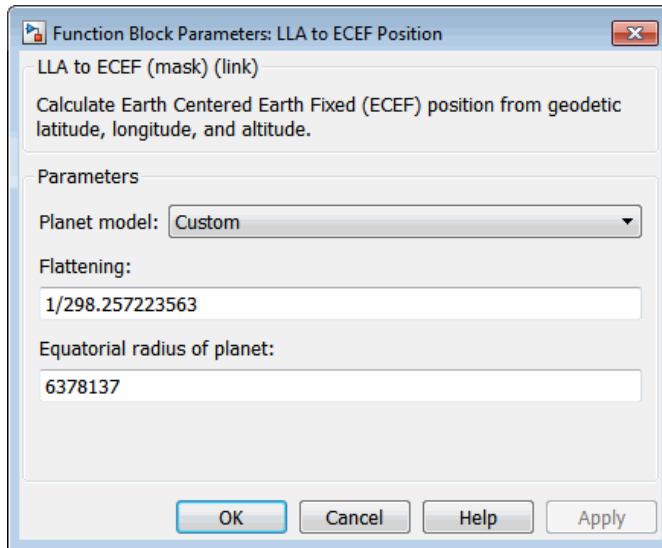
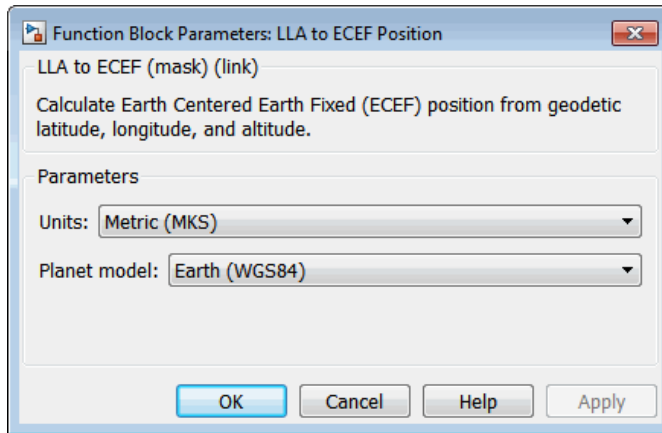
where geocentric latitude at mean sea-level and the radius at a surface point ( $r_s$ ) are defined by flattening ( $\bar{f}$ ), and equatorial radius ( $\bar{R}$ ) in the following relationships.

$$\lambda_s = \text{atan}((1 - f)^2 \tan \mu)$$

$$r_s = \sqrt{\frac{\bar{R}^2}{1 + (1 / (1 - f)^2 - 1) \sin^2 \lambda_s}}$$

# LLA to ECEF Position

## Dialog Box



### Units

Specifies the parameter and output units:

<b>Units</b>	<b>Altitude</b>	<b>Equatorial Radius</b>	<b>Position</b>
Metric (MKS)	Meters	Meters	Meters
English	Feet	Feet	Feet

This option is only available when **Planet model** is set to Earth (WGS84).

### **Planet model**

Specifies the planet model to use: Custom or Earth (WGS84).

### **Flattening**

Specifies the flattening of the planet. This option is only available with **Planet model** set to Custom.

### **Equatorial radius of planet**

Specifies the radius of the planet at its equator. The units of the equatorial radius parameter should be the same as the units for altitude. This option is only available with **Planet model** set to Custom.

## **Inputs and Outputs**

<b>Input</b>	<b>Dimension Type</b>	<b>Description</b>
First	2-by-1 vector	Contains the geodetic latitude and longitude, in degrees.
Second	Scalar	Contains the altitude above the planetary ellipsoid.

<b>Output</b>	<b>Dimension Type</b>	<b>Description</b>
First	3-by-1 vector	Contains the position in ECEF frame, in same units as altitude.

# LLA to ECEF Position

---

## **Assumptions and Limitations**

The planet is assumed to be ellipsoidal. To use a spherical planet, set the **Flattening** parameter to zero.

The implementation of the ECEF coordinate system assumes that the origin is at the center of the planet, the  $x$ -axis intersects the Greenwich meridian and the equator, the  $z$ -axis being the mean spin axis of the planet, positive to the north, and the  $y$ -axis completes the right-handed system.

## **References**

Stevens, B. L., and F. L. Lewis, *Aircraft Control and Simulation*, John Wiley & Sons, New York, 1992.

Zipfel, P. H., *Modeling and Simulation of Aerospace Vehicle Dynamics*, AIAA Education Series, Reston, Virginia, 2000.

“Atmospheric and Space Flight Vehicle Coordinate Systems,” ANSI/AIAA R-004-1992.

## **See Also**

See “About Aerospace Coordinate Systems” on page 2-12.

Direction Cosine Matrix ECEF to NED

Direction Cosine Matrix ECEF to NED to Latitude and Longitude

ECEF Position to LLA

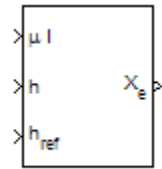
Flat Earth to LLA

Radius at Geocentric Latitude

**Purpose** Estimate flat Earth position from geodetic latitude, longitude, and altitude

**Library** Utilities/Axes Transformations

## Description



LLA to Flat Earth

The LLA to Flat Earth block converts a geodetic latitude ( $\bar{\mu}$ ), longitude ( $\bar{\iota}$ ), and altitude ( $h$ ) into a 3-by-1 vector of Flat Earth position ( $\bar{p}$ ). The flat Earth coordinate system assumes the  $z$ -axis is downward positive. The estimation begins by finding the small changes in latitude and longitude from the output latitude and longitude minus the initial latitude and longitude.

$$d\mu = \mu - \mu_0$$

$$d\iota = \iota - \iota_0$$

To convert geodetic latitude and longitude to the North and East coordinates, the estimation uses the radius of curvature in the prime vertical ( $R_N$ ) and the radius of curvature in the meridian ( $R_M$ ).  $R_N$  and  $R_M$  are defined by the following relationships:

$$R_N = \frac{R}{\sqrt{1 - (2f - f^2)\sin^2 \mu_0}}$$

$$R_M = R_N \frac{1 - (2f - f^2)}{1 - (2f - f^2)\sin^2 \mu_0}$$

where ( $R$ ) is the equatorial radius of the planet and  $f$  is the flattening of the planet.

Small changes in the North (dN) and East (dE) positions are approximated from small changes in the North and East positions by

## LLA to Flat Earth

---

$$dN = \frac{d\mu}{\operatorname{atan}\left(\frac{1}{R_M}\right)}$$
$$dE = \frac{d\iota}{\operatorname{atan}\left(\frac{1}{R_N \cos \mu_0}\right)}$$

With the conversion of the North and East coordinates to the flat Earth  $x$  and  $y$  coordinates, the transformation has the form of

$$\begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} \cos \psi & \sin \psi \\ -\sin \psi & \cos \psi \end{bmatrix} \begin{bmatrix} N \\ E \end{bmatrix}$$

where

$$(\psi)$$

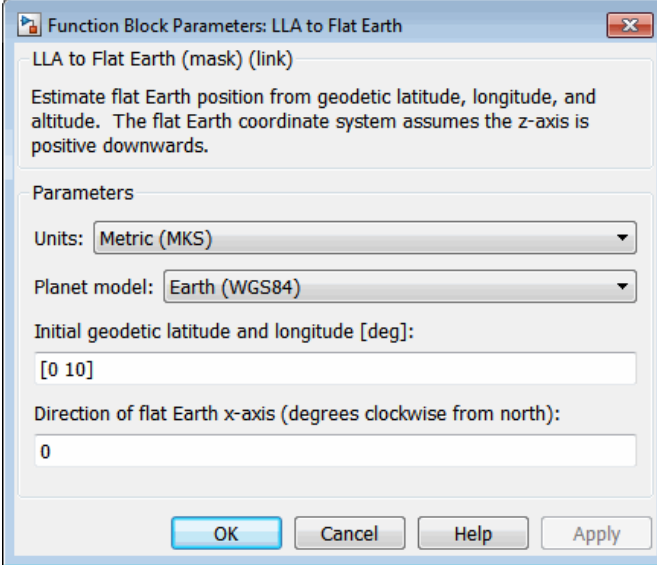
is the angle in degrees clockwise between the  $x$ -axis and north.

The flat Earth  $z$ -axis value is the negative altitude minus the reference height ( $h_{ref}$ ).

$$p_z = -h - h_{ref}$$



## Dialog Box



Function Block Parameters: LLA to Flat Earth

LLA to Flat Earth (mask) (link)

Estimate flat Earth position from geodetic latitude, longitude, and altitude. The flat Earth coordinate system assumes the z-axis is positive downwards.

Parameters

Units: Metric (MKS)

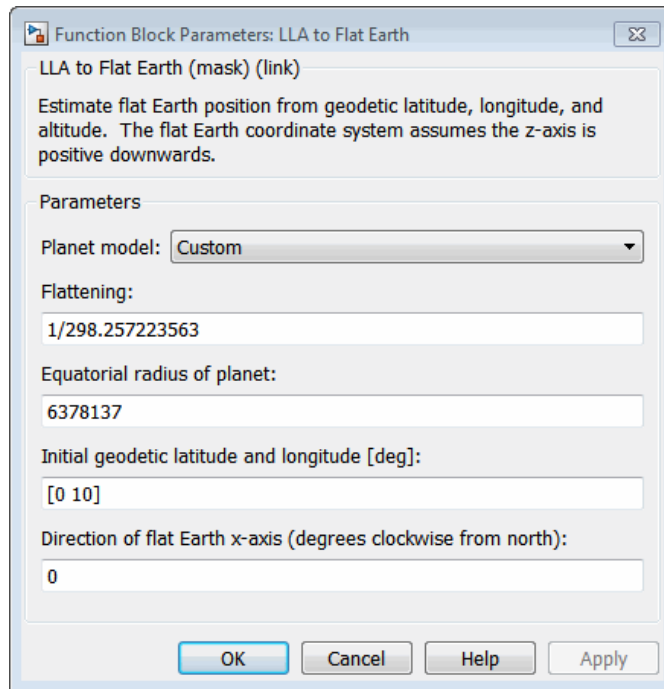
Planet model: Earth (WGS84)

Initial geodetic latitude and longitude [deg]:  
[0 10]

Direction of flat Earth x-axis (degrees clockwise from north):  
0

OK Cancel Help Apply

# LLA to Flat Earth



## Units

Specifies the parameter and output units:

Units	Position	Equatorial Radius	Altitude
Metric (MKS)	Meters	Meters	Meters
English	Feet	Feet	Feet

This option is available only when **Planet model** is set to Earth (WGS84).

## Planet model

Specifies the planet model to use: Custom or Earth (WGS84).

## Flattening

Specifies the flattening of the planet. This option is available only with **Planet model Custom**.

## Equatorial radius of planet

Specifies the radius of the planet at its equator. The units of the equatorial radius parameter should be the same as the units for flat Earth position. This option is available only with **Planet model Custom**.

## Initial geodetic latitude and longitude

Specifies the reference location, in degrees of latitude and longitude, for the origin of the estimation and the origin of the flat Earth coordinate system.

## Direction of flat Earth x-axis

Specifies angle for converting flat Earth  $x$  and  $y$  coordinates to North and East coordinates.

## Inputs and Outputs

Input	Dimension Type	Description
First	2-by-1 vector	Contains the geodetic latitude and longitude, in degrees.
Second	Scalar	Contains the altitude above the input reference altitude, in same units as flat Earth position.
Third	Scalar	Contains the reference height from the surface of the Earth to the flat Earth frame, in same units as flat Earth position. The reference height is estimated with regard to Earth frame.

Output	Dimension Type	Description
First	3-by-1 vector	Contains the position in flat Earth frame.

# LLA to Flat Earth

---

## **Assumptions and Limitations**

This estimation method assumes the flight path and bank angle are zero.

This estimation method assumes the flat Earth  $z$ -axis is normal to the Earth at the initial geodetic latitude and longitude only. This method has higher accuracy over small distances from the initial geodetic latitude and longitude, and nearer to the equator. The longitude has higher accuracy with smaller variations in latitude. Additionally, longitude is singular at the poles.

## **References**

Etkin, B. *Dynamics of Atmospheric Flight* New York: John Wiley & Sons, 1972.

Stevens, B. L., and F. L. Lewis. *Aircraft Control and Simulation*, 2nd ed. New York: John Wiley & Sons, 2003.

## **See Also**

Direction Cosine Matrix ECEF to NED

Direction Cosine Matrix ECEF to NED to Latitude and Longitude

ECEF Position to LLA

Flat Earth to LLA

Geocentric to Geodetic Latitude

LLA to ECEF Position

Radius at Geocentric Latitude

**Purpose** Compute Mach number using velocity and speed of sound

**Library** Flight Parameters

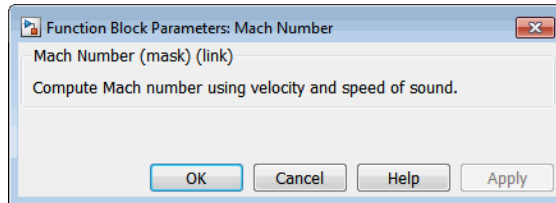
**Description** The Mach Number block computes Mach number.  
Mach number is defined as



$$Mach = \frac{\sqrt{V \cdot V}}{a}$$

where  $a$  is speed of sound and  $V$  is velocity vector.

## Dialog Box



## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the velocity vector.
Second		Contains the speed of sound.
Output	Dimension Type	Description
First	3-by-1 vector	Contains the Mach number.

**Examples** See Airframe in the aeroblk\_HL20 model for an example of this block.

**See Also** Aerodynamic Forces and Moments

# Mach Number

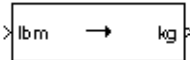
---

Dynamic Pressure

**Purpose** Convert from mass units to desired mass units

**Library** Utilities/Unit Conversions

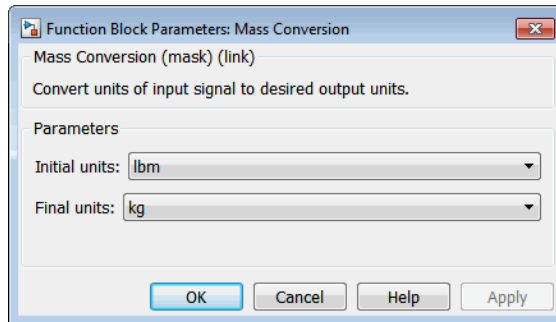
## Description



The Mass Conversion block computes the conversion factor from specified input mass units to specified output mass units and applies the conversion factor to the input signal.

The Mass Conversion block icon displays the input and output units selected from the **Initial units** and the **Final units** lists.

## Dialog Box



### Initial units

Specifies the input units.

### Final units

Specifies the output units.

The following conversion units are available:

lbr	Pound mass
kg	Kilograms
slug	Slugs

# Mass Conversion

---

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the mass in initial mass units.

Output	Dimension Type	Description
First	3-by-1 vector	Contains the mass in final mass units.

## See Also

Acceleration Conversion  
Angle Conversion  
Angular Acceleration Conversion  
Angular Velocity Conversion  
Density Conversion  
Force Conversion  
Length Conversion  
Pressure Conversion  
Temperature Conversion  
Velocity Conversion



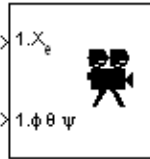
## Purpose

Create six-degrees-of-freedom multibody custom geometry block

## Library

Animation/MATLAB-Based Animation

## Description



The MATLAB Animation block creates a six-degrees-of-freedom multibody custom geometry block based on the `Aero.Animation` object. This block animates one or more vehicle geometries with  $x$ - $y$ - $z$  position and Euler angles through the specified bounding box, camera offset, and field of view. This block expects the rotation order  $z$ - $y$ - $x$  (psi, theta, phi).

To update the camera parameters in the animation, first set the parameters then close and double-click the block to reopen the MATLAB Animation window.

To access the dialog box for this block, right-click the block, then select **Mask Parameters**. Alternatively, double-click the block to display the MATLAB Animation window, then click the **Block Parameters** icon.

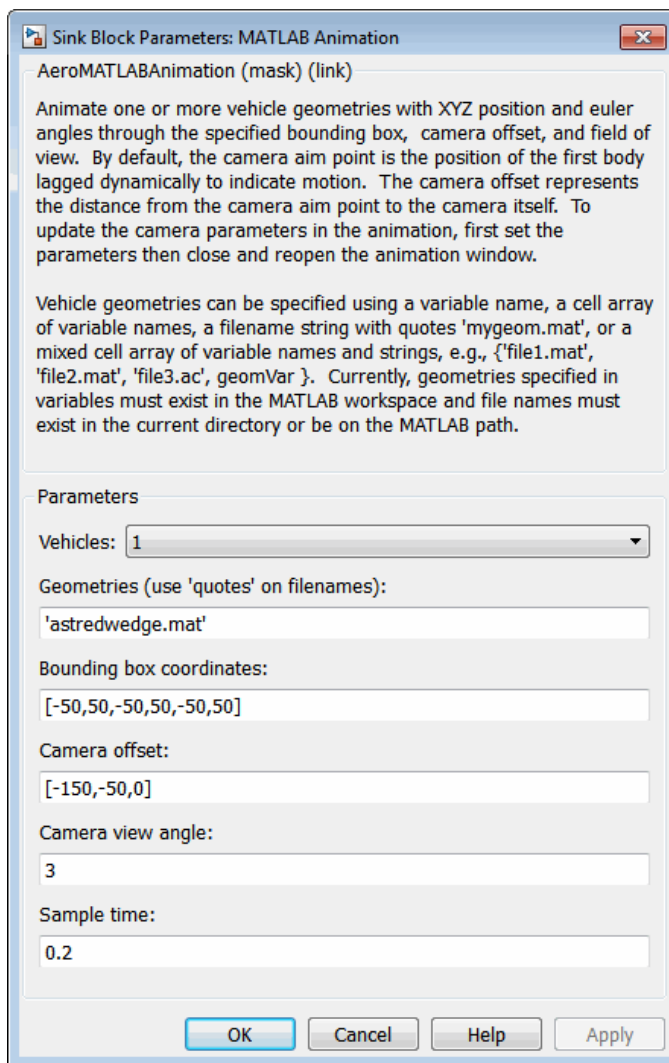
---

**Note** The underlying graphics system stores values in single precision. As a result, you might notice that motion at coordinate positions greater than approximately  $1e6$  appear unstable. This is because a single-precision number has approximately six digits of precision. The instability is due to quantization at the local value of the `eps` MATLAB function. To visualize more stable motion for coordinates beyond  $1e6$ , either offset the input data to a local zero, or scale down the coordinate values feeding the visualization.

---

# MATLAB Animation

## Dialog Box



## Vehicles

Specifies the vehicle to animate. From the list, select from 1 to 10. The block mask inputs change to reflect the number of vehicles you select. Each vehicle has its own set of inputs, denoted by the number at the beginning of the input label.

## Geometries

Specifies the vehicle geometries. You can specify these geometries using one of the following:

- Variable name, for example `geomVar`
- Cell array of variable names, for example `{geomVar, AltGeomVar}`
- String with single quotes, for example, `'astredwedge.mat'`
- Mixed cell array of variable names and strings, for example `{'file1.mat', 'file2.mat', 'file3.ac', geomVar}`

---

**Note** All specified geometries specified must exist in the MATLAB workspace and file names must exist in the current folder or be on the MATLAB path.

---

## Bounding box coordinates

Specifies the boundary coordinates for the vehicle.

This parameter is not tunable during simulation. A change to this parameter takes effect after simulation stops.

## Camera offset

Specifies the distance from the camera aim point to the camera itself.

This parameter is not tunable during simulation. A change to this parameter takes effect after simulation stops.

# MATLAB Animation

---

## Camera view angle

Specifies the camera view angle. By default, the camera aim point is the position of the first body lagged dynamically to indicate motion.

This parameter is not tunable during simulation. A change to this parameter takes effect after simulation stops.

## Sample time

Specify the sample time (-1 for inherited).

## Inputs and Outputs

Input	Dimension Type	Description
First	Vector	Contains the altitude, the crossrange position, and the downrange position of the vehicle in Earth coordinates.

Output	Dimension Type	Description
First	Vectors	Contains the Euler angles (roll, pitch, and yaw) of the vehicle.

## See Also

[Aero.Animation](#) in the Aerospace Toolbox documentation

# Moments About CG Due to Forces

## Purpose

Compute moments about center of gravity due to forces applied at a point, not center of gravity

## Library

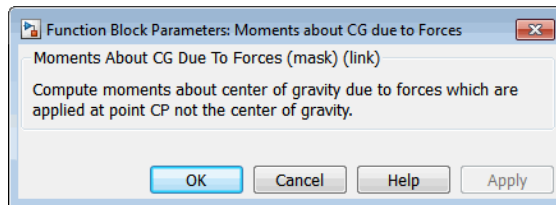
Mass Properties

## Description

The Moments about CG Due to Forces block computes moments about center of gravity due to forces that are applied at point CP, not at the center of gravity.



## Dialog Box



## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the forces applied at point CP.
Second		Contains the center of gravity.
Third		Contains the application point of forces.

Output	Dimension Type	Description
First		Contains the moments at the center of gravity in $x$ -axes, $y$ -axes and $z$ -axes.

# Moments About CG Due to Forces

---

## **See Also**

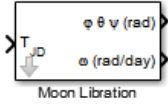
Aerodynamic Forces and Moments

Estimate Center of Gravity

**Purpose** Implement Moon librations

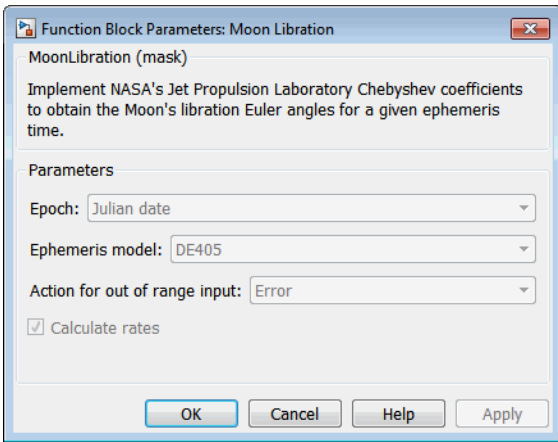
**Library** Environment/Celestial Phenomena

**Description** For a given Julian date, the Moon Libration block implements the Moon librations using Chebyshev coefficients. The block uses the Chebyshev coefficients that the NASA Jet Propulsion Laboratory provides.



**Tip** As input for the block, you can use the Aerospace Toolbox `juliandate` function to calculate the Julian date, or calculate your own Julian date, and input them using the Constant block.

## Dialog Box



**Epoch** Select one of the following:

- Julian date  
Julian date to calculate the Moon libration. When you select this option, the block has one input port.
- T0 and elapsed Julian time

# Moon Libration

---

Julian date, specified by two block inputs:

- A fixed starting epoch ( $T0$ ).
- Variable elapsed time between  $T0$  and the desired model simulation time.

$T0$  plus the variable elapsed time cannot exceed the maximum Julian date for the specified **Ephemerides**.

## Ephemeris model

Select one of the following ephemerides models defined by the Jet Propulsion Laboratory.

Ephemerides Model	Description
DE405	<p>Released in 1998. This ephemerides takes into account the Julian date range 2305424.50 (December 9, 1599 ) to 2525008.50 (February 20, 2201).</p> <p>This block implements these ephemerides with respect to the International Celestial Reference Frame version 1.0, adopted in 1998.</p>
DE421	<p>Released in 2008. This ephemerides takes into account the Julian date range 2414992.5 (December 4, 1899) to 2469808.5 (January 2, 2050).</p> <p>This block implements these ephemerides with respect to the International Celestial Reference Frame version 1.0, adopted in 1998.</p>
DE423	<p>Released in 2010. This ephemerides takes into account the Julian date range 2378480.5 (December 16, 1799) to 2524624.5 (February 1, 2200).</p> <p>This block implements these ephemerides with respect to the International Celestial Reference Frame version 2.0, adopted in 2010.</p>

## Action for out of range input

Specify the block behavior when the block inputs are out of range.



Action	Description
None	No action.
Warning	Warning in the MATLAB Command Window, model simulation continues.
Error (default)	MATLAB returns an exception, model simulation stops.

### Calculate rates

Select this check box to calculate the rate of the Moon libration and add a second block output.

### Inputs and Outputs

Input	Dimension Type	Description
First	Scalar	<ul style="list-style-type: none"> <li>Julian date (<math>T_{JD}</math>) (default) — One input. Specify a date between the minimum and maximum Julian date.</li> <li>Fixed Julian date (<math>T0_{JD}</math>) plus the elapsed Julian time (<math>\Delta T_{JD}</math>) between the fixed date and the ephemeris time. — Two inputs, where the first input is <math>T0_{JD}</math> and the second input is <math>\Delta T_{JD}</math>. <math>\Delta T_{JD}</math> must be a positive number. The sum of <math>T0_{JD}</math> and <math>\Delta T_{JD}</math> must fall between the minimum and maximum Julian date.</li> </ul> <p>The block <b>Epoch</b> parameter controls the number of block inputs.</p>

# Moon Libration

Input	Dimension Type	Description
Second (Optional)	Scalar	<p>See the <b>Ephemerides</b> parameter for the minimum and maximum Julian dates.</p> <p><math>\Delta T_{JD}</math> — Elapsed Julian time (<math>\Delta T_{JD}</math>) between the fixed date and the ephemeris time. The sum of <math>T0_{JD}</math> and <math>\Delta T_{JD}</math> must fall between the minimum and maximum Julian date.</p> <p>See the <b>Ephemerides</b> parameter for the minimum and maximum Julian date.</p>
Output	Dimension Type	Description
First	Vector	Euler angles ( $\varphi$ $\theta$ $\psi$ ) for Moon attitude. Units are radians.
Second (Optional)	Vector	Moon libration Euler angular rates ( $\omega$ ). Units are radians/day.

## References

Folkner, W. M., J. G. Williams, D. H. Boggs, “The Planetary and Lunar Ephemeris DE 421,” *IPN Progress Report 42-178*, 2009.

Vallado, D. A., *Fundamentals of Astrodynamics and Applications*, McGraw-Hill, New York, 1997.

## See Also

Earth Nutation | Planetary Ephemeris |

## External Web Sites

- [http://ssd.jpl.nasa.gov/?planet\\_eph\\_export](http://ssd.jpl.nasa.gov/?planet_eph_export)
- <http://syrte.obspm.fr/jsr/journees2010/powerpoint/folkner.pdf>

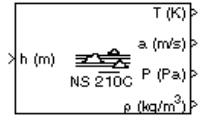
## Purpose

Implement MIL-STD-210C climatic data

## Library

Environment/Atmosphere

## Description

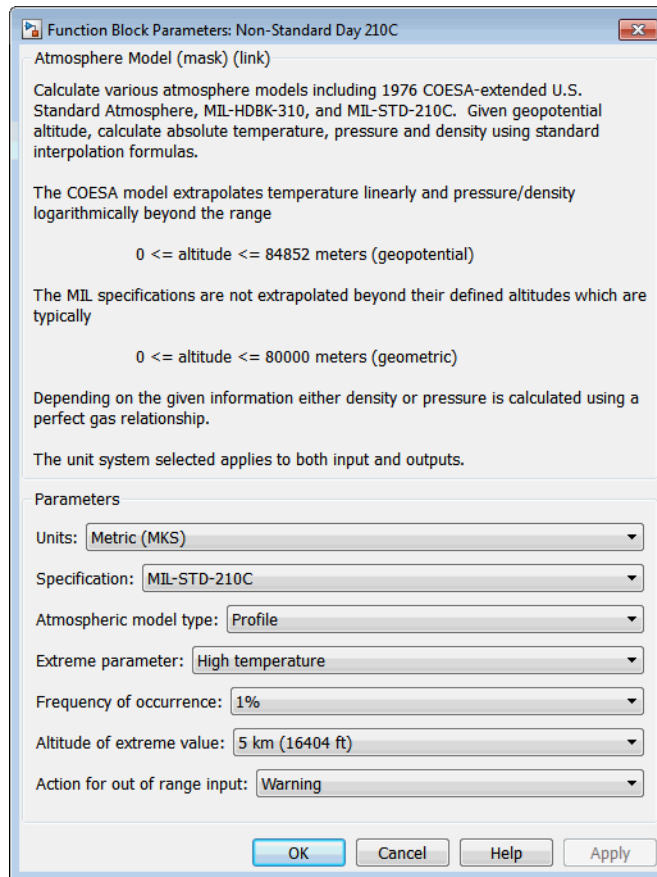


The Non-Standard Day 210C block implements a portion of the climatic data of the MIL-STD-210C worldwide air environment to 80 km (geometric or approximately 262,000 feet geometric) for absolute temperature, pressure, density, and speed of sound for the input geopotential altitude.

The Non-Standard Day 210C block icon displays the input and output units selected from the **Units** list.

# Non-Standard Day 210C

## Dialog Box



## Units

Specifies the input and output units:

<b>Units</b>	<b>Height</b>	<b>Temperature</b>	<b>Speed of Sound</b>	<b>Air Pressure</b>	<b>Air Density</b>
Metric (MKS)	Meters	Kelvin	Meters per second	Pascal	Kilograms per cubic meter
English (Velocity in ft/s)	Feet	Degrees Rankine	Feet per second	Pound force per square inch	Slug per cubic foot
English (Velocity in kts)	Feet	Degrees Rankine	Knots	Pound force per square inch	Slug per cubic foot

## Specification

Specify the atmosphere model type from one of the following atmosphere models. The default is MIL-STD-210C.

1976 COESA-extended U.S. Standard Atmosphere

This selection is linked to the COESA Atmosphere Model block. See the block reference for more information.

MIL-HDBK-310

This selection is linked to the Non-Standard Day 310 block. See the block reference for more information.

MIL-STD-210C

This selection is linked to the Non-Standard Day 210C block. See the block reference for more information.

## Atmospheric model type

Select the representation of the atmospheric data.

# Non-Standard Day 210C

---

Profile	Realistic atmospheric profiles associated with extremes at specified altitudes. Recommended for simulation of vehicles vertically traversing the atmosphere or when the total influence of the atmosphere is needed.
Envelope	Uses extreme atmospheric values at each altitude. Recommended for vehicles only horizontally traversing the atmosphere without much change in altitude.

## Extreme parameter

Select the atmospheric parameter that is the extreme value.

High temperature	Option always available
Low temperature	Option always available
High density	Option always available
Low density	Option always available
High pressure	This option is available only when Envelope is selected for <b>Atmospheric model type</b>
Low pressure	This option is available only when Envelope is selected for <b>Atmospheric model type</b>

## Frequency of occurrence

Select percent of time the values would occur.

Extreme values	This option is available only when Envelope is selected for <b>Atmospheric model type</b> .
1%	Option always available
5%	This option is available only when Envelope is selected for <b>Atmospheric model type</b> .

- 10% Option always available
- 20% This option is available only when Envelope is selected for **Atmospheric model type**.

### Altitude of extreme value

Select geometric altitude at which the extreme values occur. Applies to the profile atmospheric model only.

- 5 km (16404 ft)
- 10 km (32808 ft)
- 20 km (65617 ft)
- 30 km (98425 ft)
- 40 km (131234 ft)

### Action for out of range input

Specify if out-of-range input invokes a warning, error, or no action.

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the geopotential height.

Output	Dimension Type	Description
First		Contains the temperature.
Second		Contains the speed of sound.
Third		Contains the air pressure.
Fourth		Contains the air density.

## Assumptions and Limitations

All values are held below the geometric altitude of 0 m (0 feet) and above the geometric altitude of 80,000 meters (approximately 262,000 feet). The envelope atmospheric model has a few exceptions where values are held below the geometric altitude of 1 kilometer (approximately 3,281 feet) and above the geometric altitude of 30,000 meters (approximately

# Non-Standard Day 210C

---

98,425 feet). These exceptions arise from lack of data in MIL-STD-210C for these conditions.

In general, temperature values are interpolated linearly, and density values are interpolated logarithmically. Pressure and speed of sound are calculated using a perfect gas law. The envelope atmospheric model has a few exceptions where the extreme value is the only value provided as an output. Pressure in these cases is interpolated logarithmically. These envelope atmospheric model exceptions apply to all cases of high and low pressure, high and low temperature, and high and low density, excluding the extreme values and 1% frequency of occurrence. These exceptions arise from lack of data in MIL-STD-210C for these conditions.

Another limitation is that climatic data for the region south of 60°S latitude is excluded from consideration in MIL-STD-210C.

This block uses the metric version of data from the MIL-STD-210C specifications. Certain data within the envelope are inconsistent between metric and English versions for low density, low temperature, high temperature, low pressure, and high pressure. The most significant differences occur in the following values:

- For low density envelope data with 5% frequency, the density values in metric units are inconsistent at 4 km and 18 km and the density values in English units are inconsistent at 14 km.
- For low density envelope data with 10% frequency,
  - The density values in metric units are inconsistent at 18 km.
  - The density values in English units are inconsistent at 14 km.
- For low density envelope data with 20% frequency, the density values in English units are inconsistent at 14 km.
- For low temperature envelope data with 20% frequency, the temperature values at 20 km are inconsistent.
- For high pressure envelope data with 10% frequency, the pressure values in metric units at 8 km are inconsistent.



**Reference**

Global Climatic Data for Developing Military Products (MIL-STD-210C),  
9 January 1987, Department of Defense, Washington, D.C.

**See Also**

COESA Atmosphere Model  
ISA Atmosphere Model  
Non-Standard Day 310

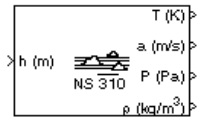
# Non-Standard Day 310

---

**Purpose** Implement MIL-HDBK-310 climatic data

**Library** Environment/Atmosphere

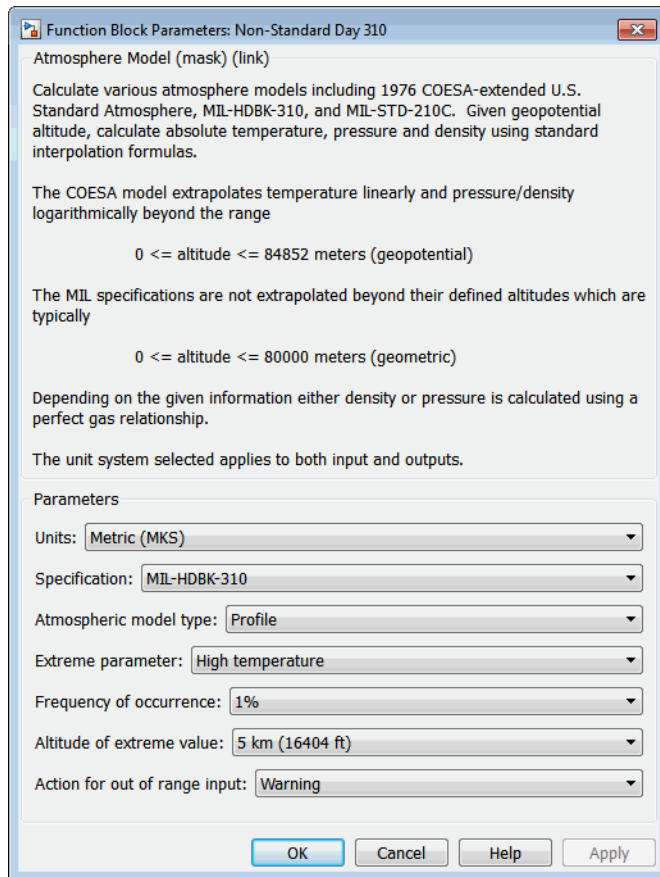
## Description



The Non-Standard Day 310 block implements a portion of the climatic data of the MIL-HDBK-310 worldwide air environment to 80 km (geometric or approximately 262,000 feet geometric) for absolute temperature, pressure, density, and speed of sound for the input geopotential altitude.

The Non-Standard Day 310 block icon displays the input and output units selected from the **Units** list.

## Dialog Box



### Units

Specifies the input and output units:

# Non-Standard Day 310

---

<b>Units</b>	<b>Height</b>	<b>Temperature</b>	<b>Speed of Sound</b>	<b>Air Pressure</b>	<b>Air Density</b>
Metric (MKS)	Meters	Kelvin	Meters per second	Pascal	Kilograms per cubic meter
English (Velocity in ft/s)	Feet	Degrees Rankine	Feet per second	Pound force per square inch	Slug per cubic foot
English (Velocity in kts)	Feet	Degrees Rankine	Knots	Pound force per square inch	Slug per cubic foot

## Specification

Specify the atmosphere model type from one of the following atmosphere models. The default is MIL-HDBK-310.

1976 COESA-extended U.S. Standard Atmosphere

This selection is linked to the COESA Atmosphere Model block. See the block reference for more information.

MIL-HDBK-310

This selection is linked to the Non-Standard Day 310 block. See the block reference for more information.

MIL-STD-210C

This selection is linked to the Non-Standard Day 210C block. See the block reference for more information.

## Atmospheric model type

Select the representation of the atmospheric data.

Profile	Realistic atmospheric profiles associated with extremes at specified altitudes. Recommended for simulation of vehicles vertically traversing the atmosphere or when the total influence of the atmosphere is needed.
Envelope	Uses extreme atmospheric values at each altitude. Recommended for vehicles only horizontally traversing the atmosphere without much change in altitude.

### Extreme parameter

Select the atmospheric parameter which is the extreme value.

High temperature	Option always available
Low temperature	Option always available
High density	Option always available
Low density	Option always available
High pressure	This option is available only when Envelope is selected for <b>Atmospheric model type</b> .
Low pressure	This option is available only when Envelope is selected for <b>Atmospheric model type</b> .

### Frequency of occurrence

Select percent of time the values would occur.

Extreme values	This option is available only when Envelope is selected for <b>Atmospheric model type</b> .
1%	Option always available
5%	This option is available only when Envelope is selected for <b>Atmospheric model type</b> .

# Non-Standard Day 310

---

- 10% Option always available
- 20% This option is available only when Envelope is selected for **Atmospheric model type**.

## Altitude of extreme value

Select geometric altitude at which the extreme values occur.  
Applies to the profile atmospheric model only.

- 5 km (16404 ft)
- 10 km (32808 ft)
- 20 km (65617 ft)
- 30 km (98425 ft)
- 40 km (131234 ft)

## Action for out of range input

Specify if out-of-range input invokes a warning, error, or no action.

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the geopotential height.

Output	Dimension Type	Description
First		Contains the temperature.
Second		Contains the speed of sound.
Third		Contains the air pressure.
Fourth		Contains the air density.

## Assumptions and Limitations

All values are held below the geometric altitude of 0 m (0 feet) and above the geometric altitude of 80,000 meters (approximately 262,000 feet). The envelope atmospheric model has a few exceptions where values are held below the geometric altitude of 1 kilometer (approximately 3,281 feet) and above the geometric altitude of 30,000 meters (approximately

98,425 feet). These exceptions arise from lack of data in MIL-HDBK-310 for these conditions.

In general, temperature values are interpolated linearly, and density values are interpolated logarithmically. Pressure and speed of sound are calculated using a perfect gas law. The envelope atmospheric model has a few exceptions where the extreme value is the only value provided as an output. Pressure in these cases is interpolated logarithmically. These envelope atmospheric model exceptions apply to all cases of high and low pressure, high and low temperature, and high and low density, excluding the extreme values and 1% frequency of occurrence. These exceptions arise from lack of data in MIL-HDBK-310 for these conditions.

Another limitation is that climatic data for the region south of 60°S latitude is excluded from consideration in MIL-HDBK-310.

This block uses the metric version of data from the MIL-STD-310 specifications. Certain data within the envelope are inconsistent between metric and English versions for low density, low temperature, high temperature, low pressure, and high pressure. The most significant differences occur in the following values:

- For low density envelope data with 5% frequency, the density values in metric units are inconsistent at 4 km and 18 km and the density values in English units are inconsistent at 14 km.
- For low density envelope data with 10% frequency,
  - The density values in metric units are inconsistent at 18 km.
  - The density values in English units are inconsistent at 14 km.
- For low density envelope data with 20% frequency, the density values in English units are inconsistent at 14 km.
- For low temperature envelope data with 20% frequency, the temperature values at 20 km are inconsistent.
- For high pressure envelope data with 10% frequency, the pressure values in metric units at 8 km are inconsistent.

# Non-Standard Day 310

---

## **Reference**

Global Climatic Data for Developing Military Products  
(MIL-HDBK-310), 23 June 1997, Department of Defense, Washington,  
D.C.

## **See Also**

COESA Atmosphere Model  
ISA Atmosphere Model  
Non-Standard Day 210C



# Nonlinear Second-Order Actuator

## Purpose

Implement second-order actuator with rate and deflection limits

## Library

Actuators

## Description



The Second Order Nonlinear Actuator block outputs the actual actuator position using the input demanded actuator position and other dialog box parameters that define the system.

## Dialog Box

Function Block Parameters: Nonlinear Second-Order Actuator

NonlinearSecondOrderActuator (mask) (link)

Implement an actuator model with saturation, rate limits, and a second-order integrator.

Parameters

Natural frequency:  
1

Damping ratio:  
0.3

Maximum deflection:  
 $20 \cdot \pi / 180$

Minimum deflection:  
 $-20 \cdot \pi / 180$

Rate limit:  
 $500 \cdot \pi / 180$

Initial position:  
0

Initial velocity:  
0

OK Cancel Help Apply

# Nonlinear Second-Order Actuator

---

## **Natural frequency**

The natural frequency of the actuator. The units of natural frequency are radians per second.

## **Damping ratio**

The damping ratio of the actuator. A dimensionless parameter.

## **Maximum deflection**

The largest actuator position allowable. The units of maximum deflection must be the same as the units of demanded actuator position.

## **Minimum deflection**

The smallest actuator position allowable. The units of minimum deflection must be the same as the units of demanded actuator position.

## **Rate limit**

The fastest speed allowable for actuator motion. The units of maximum rate must be the units of demanded actuator position per second.

## **Initial position**

The initial position of the actuator. The units of initial position must be the same as the units of demanded actuator position.

If the specified value is less than the value of **Minimum deflection**, the block sets the value of **Minimum deflection** as the initial position value. If the specified value is greater than the value of **Maximum deflection**, the block sets the value of **Maximum deflection** as the initial position value.

## **Initial velocity**

The initial velocity of the actuator. The units of initial velocity must be the same as the units of demanded actuator position per second.

If the absolute value of the specified value is greater than the absolute value of **Rate Limit**, this block sets the value of **Rate Limit** as the initial velocity value.

# Nonlinear Second-Order Actuator

---

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the demanded actuator position.

Output	Dimension Type	Description
First		Contains the actual actuator position.

## Examples

See the Airframe & Autopilot subsystem in the `aero_guidance` model and the Actuators subsystem in the `aeroblk_HL20` model.

## See Also

Linear Second-Order Actuator

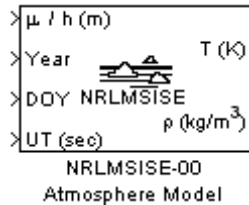
# NRLMSISE-00 Atmosphere Model

---

**Purpose** Implement mathematical representation of 2001 United States Naval Research Laboratory Mass Spectrometer and Incoherent Scatter Radar Exosphere

**Library** Environment/Atmosphere

**Description** The NRLMSISE-00 Atmosphere Model block implements the mathematical representation of the 2001 United States Naval Research Laboratory Mass Spectrometer and Incoherent Scatter Radar Exosphere (NRLMSISE-00). This block calculates the neutral atmosphere empirical model from the surface to lower exosphere (0 to 1,000,000 meters). When configuring the block for this calculation, you can also take into account the anomalous oxygen, which can affect the satellite drag above 500,000 meters.

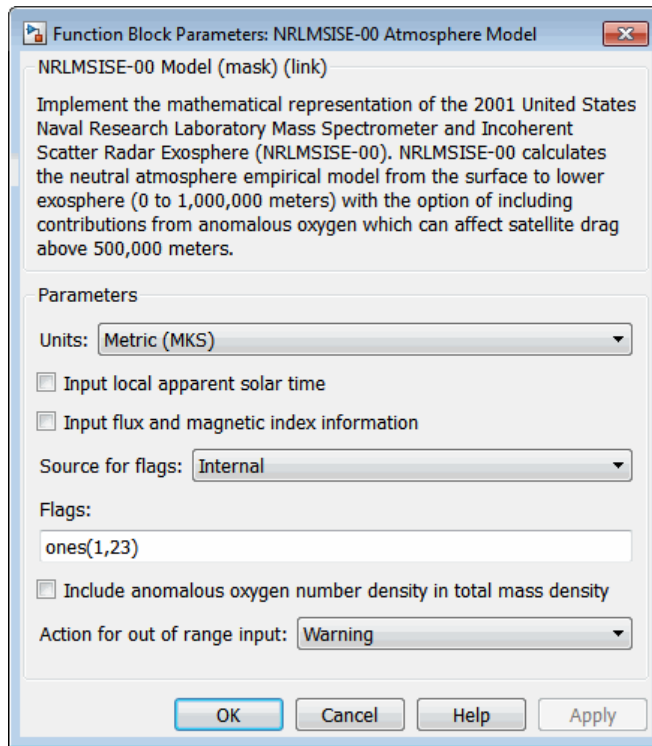


---

**Note** This block is valid only for altitudes between 0 and 1,000,000 meters (1,000 kilometers).

---

## Dialog Box



### Units

Specifies the input and output units:

Units	Temperature	Height	Density
Metric (MKS)	Kelvin	Meters	kg/m <sup>3</sup> , some density outputs 1/m <sup>3</sup>
English	Rankine	Feet	lbm/ft <sup>3</sup> , some density outputs 1/ft <sup>3</sup>

# NRLMSISE-00 Atmosphere Model

---

## Input local apparent solar time

Select this check box to input the local apparent solar time, in hours. Otherwise, the block inputs the default value.

## Input flux and magnetic index information

Select this check box to input the 81-day average of F10.7, the daily F10.7 flux for the previous day, and the array of 7 magnetic index information (see the `aph` argument in the Aerospace Toolbox `atmosnrlmsise00` function). Otherwise, the block inputs the default value.

## Source for flags

Specify the variation flag source. If you specify `External`, you must enter the variation flag as an array of 23. If you specify `Internal`, the flag source is internal to the block.

## Flags

Specify the variation flag as an array of 23. This parameter applies only when **Source for flags** has a value of `Internal`. You can specify one of the following values for a field. The default value for each field is 1.

- 0.0

Removes that value's effect on the output.

- 1.0

Applies the main and the cross-term effects of that value on the output.

- 2.0

Applies only the cross-term effect of that value on the output. The array has the following fields.

Field	Description
Flags (1)	F10.7 effect on mean
Flags (2)	Independent of time

# NRLMSISE-00 Atmosphere Model

<b>Field</b>	<b>Description</b>
Flags (3)	Symmetrical annual
Flags (4)	Symmetrical semiannual
Flags (5)	Asymmetrical annual
Flags (6)	Asymmetrical semiannual
Flags (7)	Diurnal
Flags (8)	Semidiurnal
Flags (9)	Daily AP. If you set this field to -1, the block uses the entire matrix of magnetic index information (APH) instead of APH(: , 1)
Flags (10)	All UT, longitudinal effects
Flags (11)	Longitudinal
Flags (12)	UT and mixed UT, longitudinal
Flags (13)	Mixed AP, UT, longitudinal
Flags (14)	Terdiurnal
Flags (15)	Departures from diffusive equilibrium
Flags (16)	All exospheric temperature variations
Flags (17)	All variations from 120,000 meter temperature (TLB)
Flags (18)	All lower thermosphere (TN1) temperature variations
Flags (19)	All 120,000 meter gradient (S) variations
Flags (20)	All upper stratosphere (TN2) temperature variations
Flags (21)	All variations from 120,000 meter values (ZLB)

# NRLMSISE-00 Atmosphere Model

---

Field	Description
Flags (22)	All lower mesosphere temperature (TN3) variations
Flags (23)	Turbopause scale height variations

## **Include anomalous oxygen number density in total mass density**

Select this check box to take into account the anomalous oxygen when calculating the neutral atmosphere empirical model from the surface to lower exosphere (0 to 1,000,000 meters). Taking into account this number can affect the satellite drag above 500,000 meters.

## **Action for out of range input**

Specify if out-of-range input invokes a warning, error, or no action.

## **Inputs and Outputs**

Input	Dimension Type	Description
First	Three-element matrix	Contains latitudes, in degrees, longitude, in degrees, and altitude, in selected length units.
Second	Array	Contains N years.
Third	Array	Contains N days of a year (1 to 365 (or 366)).
Fourth	Array	Contains N seconds in a day, in universal time (UT).
Fifth (Optional)	Array	Contains N local apparent solar time, in hours.
Sixth (Optional)	Array	Contains N 81-day average of F10.7 flux, centered on day of year (doy).
Seventh (Optional)	Array	Contains N daily F10.7 flux for previous day.



<b>Input</b>	<b>Dimension Type</b>	<b>Description</b>
Eight (Optional)	N-by-7 array	Contains N-by-7 of magnetic index information.
Ninth (Optional)	Array of 23	Contains flags to enable or disable particular variations for the outputs. See following table.

These flags, associated with the ninth input, enable or disable particular variations for the outputs.

<b>Field</b>	<b>Description</b>
Flags(1)	F10.7 effect on mean
Flags(2)	Independent of time
Flags(3)	Symmetrical annual
Flags(4)	Symmetrical semiannual
Flags(5)	Asymmetrical annual
Flags(6)	Asymmetrical semiannual
Flags(7)	Diurnal
Flags(8)	Semidiurnal
Flags(9)	Daily AP. If you set this field to -1, the block uses the entire matrix of magnetic index information (APH) instead of APH(:,1)
Flags(10)	All UT, longitudinal effects
Flags(11)	Longitudinal
Flags(12)	UT and mixed UT, longitudinal
Flags(13)	Mixed AP, UT, longitudinal
Flags(14)	Terdiurnal

# NRLMSISE-00 Atmosphere Model

Field	Description
Flags(15)	Departures from diffusive equilibrium
Flags(16)	All exospheric temperature variations
Flags(17)	All variations from 120,000 meter temperature (TLB)
Flags(18)	All lower thermosphere (TN1) temperature variations
Flags(19)	All 120,000 meter gradient (S) variations
Flags(20)	All upper stratosphere (TN2) temperature variations
Flags(21)	All variations from 120,000 meter values (ZLB)
Flags(22)	All lower mesosphere temperature (TN3) variations
Flags(23)	Turbopause scale height variations

The outputs are:

Output	Dimension Type	Description
First	Array	Contains N-by-2 values of temperature, in selected temperature units. The first column is exospheric temperature, the second column is temperature at altitude.
Second	Array	Contains N-by-9 values of densities in selected density units. See the following table:

These densities are associated with the second output.

Density	Description
Density(1)	Density of He
Density(2)	Density of O
Density(3)	Density of N2

# NRLMSISE-00 Atmosphere Model

Density	Description
Density(4)	Density of O2
Density(5)	Density of Ar
Density(6)	Total mass density Density (6), total mass density, is defined as the sum of the mass densities of He, O, N2, O2, Ar, H, and N. Optionally, Density (6) can include the mass density of anomalous oxygen making Density (6), the effective total mass density for drag.
Density(7)	Density of H
Density(8)	Density of N
Density(9)	Anomalous oxygen number density

## Assumptions and Limitations

The F107 and F107A values that are used to generate the model correspond to the 10.7 cm radio flux at the actual distance of the Earth from the Sun rather than the radio flux at 1 AU. The following site provides both classes of values:

[ftp://ftp.ngdc.noaa.gov/STP/SOLAR\\_DATA/SOLAR\\_RADIO/FLUX/](ftp://ftp.ngdc.noaa.gov/STP/SOLAR_DATA/SOLAR_RADIO/FLUX/)

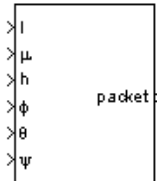
# Pack net\_fdm Packet for FlightGear

---

**Purpose** Generate net\_fdm packet for FlightGear

**Library** Animation/Flight Simulator Interfaces

## Description



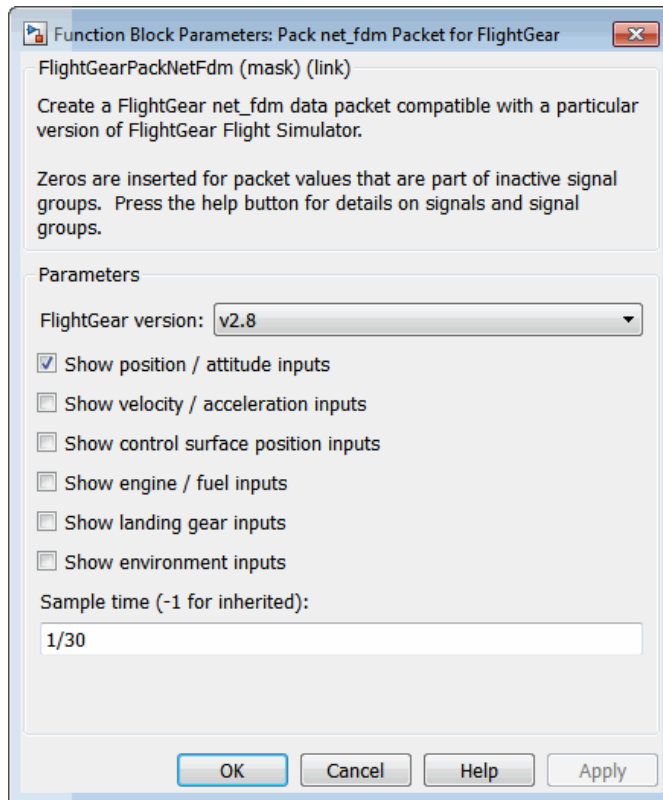
The Pack net\_fdm Packet for FlightGear block creates, from separate inputs, a FlightGear net\_fdm data packet compatible with a particular version of FlightGear flight simulator. All the signals supported by the FlightGear net\_fdm data packet are supported by this block. The signals are arranged into six groups. Any group can be turned on or off. Zeros are inserted for packet values that are part of inactive signal groups.

See “Inputs and Outputs” on page 4-416 for details on signals and signal groups.

Supported FlightGear versions: v0.9.3, v0.9.8/0.9.8a, v0.9.9, v0.9.10, v1.0, v1.9.1, v2.0, v2.4, v2.6, v2.8.

# Pack net\_fdm Packet for FlightGear

## Dialog Box



### FlightGear version

Select your FlightGear software version.

Supported FlightGear versions: v0.9.3, v0.9.8/0.9.8a, v0.9.9, v0.9.10, v1.0, v1.9.1, v2.0, v2.4, v2.6, v2.8.

### Show position/altitude inputs

Select this check box to include the position and altitude inputs (signal group 1) into the FlightGear net\_fdm data packet.

# Pack net\_fdm Packet for FlightGear

---

## Show velocity/acceleration inputs

Select this check box to include the velocity and acceleration inputs (signal group 2) into the FlightGear net\_fdm data packet.

## Show control surface position inputs

Select this check box to include the control surface position inputs (signal group 3) into the FlightGear net\_fdm data packet.

## Show engine/fuel inputs

Select this check box to include the engine and fuel inputs (signal group 4) into the FlightGear net\_fdm data packet.

## Show landing gear inputs

Select this check box to include the landing gear inputs (signal group 5) into the FlightGear net\_fdm data packet.

## Show environment inputs

Select this check box to include the environment inputs (signal group 6) into the FlightGear net\_fdm data packet.

## Sample time

Specify the sample time (-1 for inherited).

## Inputs and Outputs

### Input Signals Supported for FlightGear 0.9.3

This table lists all the input signals supported for Version 0.9.3:

#### Signal Group 1: ShowPositionAttitudeInputs

Name	Units	Type	Width	Description
$l$	rad	double	1	Geodetic longitude
$\mu$	rad	double	1	Geodetic altitude
$h$	m	double	1	Altitude above sea level
$\Phi$	rad	single	1	Roll

# Pack net\_fdm Packet for FlightGear

## Signal Group 1: ShowPositionAttitudeInputs (Continued)

Name	Units	Type	Width	Description
$\Theta$	rad	single	1	Pitch
$\Psi$	rad	single	1	Yaw or true heading

## Signal Group 2: ShowVelocityAccelerationInputs

Name	Units	Type	Width	Description
$d\Phi/dt$	rad/sec	single	1	Roll rate
$d\Phi/dt$	rad/sec	single	1	Pitch rate
$d\psi/dt$	rad/sec	single	1	Yaw rate
vcas	kts	single	1	Calibrated airspeed
climb_rate	ft/sec	single	1	Climb rate
v_north	ft/sec	single	1	North velocity in local/body frame
v_east	ft/sec	single	1	East velocity in local/body frame
v_down	ft/sec	single	1	Down/vertical velocity in local/body frame
v_wind_body_north	ft/sec	single	1	Body north velocity relative to local airmass
v_wind_body_east	ft/sec	single	1	Body east velocity relative to local airmass
v_wind_body_down	ft/sec	single	1	Body down/vertical velocity relative to local airmass

# Pack net\_fdm Packet for FlightGear

## Signal Group 2: ShowVelocityAccelerationInputs (Continued)

Name	Units	Type	Width	Description
stall_warning	—	single	1	0.0–1.0, indicating the amount of stall
A_X_pilot	ft/sec <sup>2</sup>	single	1	X acceleration in body frame
A_Y_pilot	ft/sec <sup>2</sup>	single	1	Y acceleration in body frame
A_Z_pilot	ft/sec <sup>2</sup>	single	1	Z acceleration in body frame

## Signal Group 3: ShowControlSurfacePositionInputs

Name	Units	Type	Width	Description
elevator	geometry-specific units	single	1	Elevator position
flaps	geometry-specific units	single	1	Flaps position
left_aileron	geometry-specific units	single	1	Left aileron position
right_aileron	geometry-specific units	single	1	Right aileron position
rudder	geometry-specific units	single	1	Rudder position
speedbrake	geometry-specific units	single	1	Speed brake position
spoilers	geometry-specific units	single	1	Spoilers position



# Pack net\_fdm Packet for FlightGear

## Signal Group 4: ShowEngineFuelInputs

Name	Units	Type	Width	Description
num_engines	—	int32	1	Number of valid engines
eng_state	enum	int32	4	Engine state (0=off, 1=cranking, 2=running)
rpm	rev/min	single	4	Engine RPM
fuel_flow	gal/hr	single	4	Fuel flow
EGT	°F	single	4	Exhaust gas temp
oil_temp	°F	single	4	Oil temp
oil_px	lbf/in <sup>2</sup>	single	4	Oil pressure
num_tanks	—	int32	1	Max number of fuel tanks
fuel_quantity	—	single	4	Amount of fuel in tanks (0–1 fraction)

## Signal Group 5: ShowLandingGearInputs

Name	Units	Type	Width	Description
num_wheels	—	int32	1	Maximum number of wheels
wow	—	boolean	3	Weight on wheels signal (1=wheel is on ground)
gear_pos	—	single	3	Landing gear position (0-1, indicating amount deployed)
gear_steer	—	single	3	Landing gear steering angle
gear_compression	—	single	3	Landing gear compression

# Pack net\_fdm Packet for FlightGear

---

## Signal Group 6: ShowEnvironmentInputs

Name	Units	Type	Width	Description
agl	m	single	1	Above ground level
cur_time	sec	int32	1	Current UNIX time
warp	sec	int32	1	Offset in seconds to UNIX time
visibility	m	single	1	Visibility in meters (for visual effects)

## Input Signals Supported for FlightGear 0.9.8/0.9.8a

This table lists all the input signals supported for Versions 0.9.8/0.9.8a:

## Signal Group 1: ShowPositionAttitudeInputs

Name	Units	Type	Width	Description
$l$	rad	double	1	Geodetic longitude
$\mu$	rad	double	1	Geodetic altitude
$h$	m	double	1	Altitude above sea level
$\Phi$	rad	single	1	Roll
$\Theta$	rad	single	1	Pitch
$\psi$	rad	single	1	Yaw or true heading

## Signal Group 2: ShowVelocityAccelerationInputs

Name	Units	Type	Width	Description
$\alpha$	rad	single	1	Angle of attack
$\beta$	rad	single	1	sideslip angle

# Pack net\_fdm Packet for FlightGear

## Signal Group 2: ShowVelocityAccelerationInputs (Continued)

Name	Units	Type	Width	Description
d $\Phi$ /dt	rad/sec	single	1	Roll rate
d $\Theta$ /dt	rad/sec	single	1	Pitch rate
d $\Psi$ /dt	rad/sec	single	1	Yaw rate
vcas	kts	single	1	Calibrated airspeed
climb_rate	ft/sec	single	1	Climb rate
v_north	ft/sec	single	1	North velocity in local/body frame
v_east	ft/sec	single	1	East velocity in local/body frame
v_down	ft/sec	single	1	Down/vertical velocity in local/body frame
v_wind_body_north	ft/sec	single	1	Body north velocity relative to local airmass
v_wind_body_east	ft/sec	single	1	Body east velocity relative to local airmass
v_wind_body_down	ft/sec	single	1	Body down/vertical velocity relative to local airmass
A_X_pilot	ft/sec <sup>2</sup>	single	1	X acceleration in body frame
A_Y_pilot	ft/sec <sup>2</sup>	single	1	Y acceleration in body frame
A_Z_pilot	ft/sec <sup>2</sup>	single	1	Z acceleration in body frame
stall_warning	—	single	1	0.0–1.0, indicating the amount of stall
slip_deg	deg	single	1	Slip ball deflection

# Pack net\_fdm Packet for FlightGear

## Signal Group 3: ShowControlSurfacePositionInputs

Name	Units	Type	Width	Description
elevator	geometry-specific units	single	1	Elevator position
elevator_trim_tab	geometry-specific units	single	1	Elevator trim position
left_flap	geometry-specific units	single	1	Left flap position
right_flap	geometry-specific units	single	1	Right flap position
left_aileron	geometry-specific units	single	1	Left aileron position
right_aileron	geometry-specific units	single	1	Right aileron position
rudder	geometry-specific units	single	1	Rudder position
nose_wheel	geometry-specific units	single	1	Nose wheel position
speedbrake	geometry-specific units	single	1	Speed brake position
spoilers	geometry-specific units	single	1	Spoilers position

## Signal Group 4: ShowEngineFuelInputs

Name	Units	Type	Width	Description
num_engines	—	int32	1	Number of valid engines
eng_state	enum	int32	4	Engine state (0=off, 1=cranking, 2=running)

# Pack net\_fdm Packet for FlightGear

## Signal Group 4: ShowEngineFuelInputs (Continued)

Name	Units	Type	Width	Description
rpm	rev/min	single	4	Engine RPM
fuel_flow	gal/hr	single	4	Fuel flow
EGT	°F	single	4	Exhaust gas temp
cht	°F	single	4	Cylinder head temperature
mp_osi	psi	single	4	Manifold pressure
tit	°F	single	4	Turbine inlet temperature
oil_temp	°F	single	4	Oil temp
oil_px	lbf/in <sup>2</sup>	single	4	Oil pressure
num_tanks	—	int32	1	Max number of fuel tanks
fuel_quantity	—	single	4	Amount of fuel in tanks (0–1 fraction)

## Signal Group 5: ShowLandingGearInputs

Name	Units	Type	Width	Description
num_wheels	—	int32	1	Maximum number of wheels
wow	—	boolean	3	Weight on wheels signal (1=wheel is on ground)
gear_pos	—	single	3	Landing gear position (0–1, indicating amount deployed)

# Pack net\_fdm Packet for FlightGear

---

## Signal Group 5: ShowLandingGearInputs (Continued)

Name	Units	Type	Width	Description
gear_steer	—	single	3	Landing gear steering angle
gear_compression	—	single	3	Landing gear compression

## Signal Group 6: ShowEnvironmentInputs

Name	Units	Type	Width	Description
agl	m	single	1	Above ground level
cur_time	sec	int32	1	Current UNIX time
warp	sec	int32	1	Offset in seconds to UNIX time
visibility	m	single	1	Visibility in meters (for visual effects)

## Input Signals Supported for FlightGear 0.9.9

This table lists all the input signals supported for Version 0.9.9:

## Signal Group 1: ShowPositionAttitudeInputs

Name	Units	Type	Width	Description
$l$	rad	double	1	Geodetic longitude
$\mu$	rad	double	1	Geodetic latitude
$h$	m	double	1	Altitude above sea level
$\Phi$	rad	single	1	Roll
$\Theta$	rad	single	1	Pitch
$\psi$	rad	single	1	Yaw or true heading

## Signal Group 2: ShowVelocityAccelerationInputs

Name	Units	Type	Width	Description
$\alpha$	rad	single	1	Angle of attack
$\beta$	rad	single	1	sideslip angle
d $\Phi$ /dt	rad/sec	single	1	Roll rate
d $\Theta$ /dt	rad/sec	single	1	Pitch rate
d $\Psi$ /dt	rad/sec	single	1	Yaw rate
vcas	kts	single	1	Calibrated airspeed
climb_rate	ft/sec	single	1	Climb rate
v_north	ft/sec	single	1	North velocity in local/body frame
v_east	ft/sec	single	1	East velocity in local/body frame
v_down	ft/sec	single	1	Down/vertical velocity in local/body frame
v_wind_body_north	ft/sec	single	1	Body north velocity relative to local airmass
v_wind_body_east	ft/sec	single	1	Body east velocity relative to local airmass
v_wind_body_down	ft/sec	single	1	Body down/vertical velocity relative to local airmass
A_X_pilot	ft/sec <sup>2</sup>	single	1	X acceleration in body frame
A_Y_pilot	ft/sec <sup>2</sup>	single	1	Y acceleration in body frame
A_Z_pilot	ft/sec <sup>2</sup>	single	1	Z acceleration in body frame

# Pack net\_fdm Packet for FlightGear

---

## Signal Group 2: ShowVelocityAccelerationInputs (Continued)

Name	Units	Type	Width	Description
stall_warning	—	single	1	0.0–1.0, indicating the amount of stall
slip_deg	deg	single	1	Slip ball deflection

## Signal Group 3: ShowControlSurfacePositionInputs

Name	Units	Type	Width	Description
elevator	geometry-specific units	single	1	Elevator position
elevator_trim_tab	geometry-specific units	single	1	Elevator trim position
left_flap	geometry-specific units	single	1	Left flap position
right_flap	geometry-specific units	single	1	Right flap position
left_aileron	geometry-specific units	single	1	Left aileron position
right_aileron	geometry-specific units	single	1	Right aileron position
rudder	geometry-specific units	single	1	Rudder position
nose_wheel	geometry-specific units	single	1	Nose wheel position
speedbrake	geometry-specific units	single	1	Speed brake position
spoilers	geometry-specific units	single	1	Spoilers position



# Pack net\_fdm Packet for FlightGear

## Signal Group 4: ShowEngineFuelInputs

Name	Units	Type	Width	Description
num_engines	—	uint32	1	Number of valid engines
eng_state	enum	uint32	4	Engine state (0=off, 1=cranking, 2=running)
rpm	rev/min	single	4	Engine RPM
fuel_flow	gal/hr	single	4	Fuel flow
EGT	°F	single	4	Exhaust gas temp
cht	°F	single	4	Cylinder head temperature
mp_osi	psi	single	4	Manifold pressure
tit	°F	single	4	Turbine inlet temperature
oil_temp	°F	single	4	Oil temp
oil_px	lbf/in <sup>2</sup>	single	4	Oil pressure
num_tanks	—	uint32	1	Max number of fuel tanks
fuel_quantity	—	single	4	Amount of fuel in tanks (0–1 fraction)

## Signal Group 5: ShowLandingGearInputs

Name	Units	Type	Width	Description
num_wheels	—	uint32	1	Maximum number of wheels
wow	—	uint32	3	Weight on wheels signal (1=wheel is on ground)

# Pack net\_fdm Packet for FlightGear

## Signal Group 5: ShowLandingGearInputs (Continued)

Name	Units	Type	Width	Description
gear_pos	—	single	3	Landing gear position (0-1, indicating amount deployed)
gear_steer	—	single	3	Landing gear steering angle
gear_compression	—	single	3	Landing gear compression

## Signal Group 6: ShowEnvironmentInputs

Name	Units	Type	Width	Description
agl	m	single	1	Above ground level
cur_time	sec	uint32	1	Current UNIX time
warp	sec	int32	1	Offset in seconds to UNIX time
visibility	m	single	1	Visibility in meters (for visual effects)

## Input Signals Supported for FlightGear 0.9.10/1.0/1.9.1/2.0/2.4/2.6/2.8

This table lists all the input signals supported for Version 0.9.10, 1.0, 1.9.1, 2.0, 2.4, 2.6, and 2.8:

## Signal Group 1: ShowPositionAttitudeInputs

Name	Units	Type	Width	Description
$l$	rad	double	1	Geodetic longitude
$\mu$	rad	double	1	Geodetic latitude

# Pack net\_fdm Packet for FlightGear

## Signal Group 1: ShowPositionAttitudeInputs (Continued)

Name	Units	Type	Width	Description
$h$	m	double	1	Altitude above sea level
$\Phi$	rad	single	1	Roll
$\Theta$	rad	single	1	Pitch
$\psi$	rad	single	1	Yaw or true heading

## Signal Group 2: ShowVelocityAccelerationInputs

Name	Units	Type	Width	Description
$\alpha$	rad	single	1	Angle of attack
$\beta$	rad	single	1	sideslip angle
$d\Phi/dt$	rad/sec	single	1	Roll rate
$d\Theta/dt$	rad/sec	single	1	Pitch rate
$d\psi/dt$	rad/sec	single	1	Yaw rate
vcas	kts	single	1	Calibrated airspeed
climb_rate	ft/sec	single	1	Climb rate
v_north	ft/sec	single	1	North velocity in local/body frame
v_east	ft/sec	single	1	East velocity in local/body frame
v_down	ft/sec	single	1	Down/vertical velocity in local/body frame
v_wind_body_north	ft/sec	single	1	Body north velocity relative to local airmass
v_wind_body_east	ft/sec	single	1	Body east velocity relative to local airmass

# Pack net\_fdm Packet for FlightGear

## Signal Group 2: ShowVelocityAccelerationInputs (Continued)

Name	Units	Type	Width	Description
v_wind_body_down	ft/sec	single	1	Body down/vertical velocity relative to local airmass
A_X_pilot	ft/sec <sup>2</sup>	single	1	X acceleration in body frame
A_Y_pilot	ft/sec <sup>2</sup>	single	1	Y acceleration in body frame
A_Z_pilot	ft/sec <sup>2</sup>	single	1	Z acceleration in body frame
stall_warning	—	single	1	0.0–1.0, indicating the amount of stall
slip_deg	deg	single	1	Slip ball deflection

## Signal Group 3: ShowControlSurfacePositionInputs

Name	Units	Type	Width	Description
elevator	geometry-specific units	single	1	Elevator position
elevator_trim_tab	geometry-specific units	single	1	Elevator trim position
left_flap	geometry-specific units	single	1	Left flap position
right_flap	geometry-specific units	single	1	Right flap position
left_aileron	geometry-specific units	single	1	Left aileron position
right_aileron	geometry-specific units	single	1	Right aileron position

# Pack net\_fdm Packet for FlightGear

## Signal Group 3: ShowControlSurfacePositionInputs (Continued)

Name	Units	Type	Width	Description
rudder	geometry-specific units	single	1	Rudder position
nose_wheel	geometry-specific units	single	1	Nose wheel position
speedbrake	geometry-specific units	single	1	Speed brake position
spoilers	geometry-specific units	single	1	Spoilers position

## Signal Group 4: ShowEngineFuelInputs

Name	Units	Type	Width	Description
num_engines	—	uint32	1	Number of valid engines
eng_state	enum	uint32	4	Engine state (0=off, 1=cranking, 2=running)
rpm	rev/min	single	4	Engine RPM
fuel_flow	gal/hr	single	4	Fuel flow
fuel_px	psi	single	4	Fuel pressure
EGT	°F	single	4	Exhaust gas temp
cht	°F	single	4	Cylinder head temperature
mp_osi	psi	single	4	Manifold pressure
tit	°F	single	4	Turbine inlet temperature
oil_temp	°F	single	4	Oil temp
oil_px	lbf/in <sup>2</sup>	single	4	Oil pressure

# Pack net\_fdm Packet for FlightGear

---

## Signal Group 4: ShowEngineFuelInputs (Continued)

Name	Units	Type	Width	Description
num_tanks	—	uint32	1	Max number of fuel tanks
fuel_quantity	—	single	4	Amount of fuel in tanks (0–1 fraction)

## Signal Group 5: ShowLandingGearInputs

Name	Units	Type	Width	Description
num_wheels	—	uint32	1	Maximum number of wheels
wow	—	uint32	3	Weight on wheels signal (1=wheel is on ground)
gear_pos	—	single	3	Landing gear position (0–1, indicating amount deployed)
gear_steer	—	single	3	Landing gear steering angle
gear_compression	—	single	3	Landing gear compression

## Signal Group 6: ShowEnvironmentInputs

Name	Units	Type	Width	Description
agl	m	single	1	Above ground level
cur_time	sec	uint32	1	Current UNIX time

## Signal Group 6: ShowEnvironmentInputs (Continued)

Name	Units	Type	Width	Description
warp	sec	int32	1	Offset in seconds to UNIX time
visibility	m	single	1	Visibility in meters (for visual effects)

### Output Signal

The output signal is the FlightGear net\_fdm data packet.

### Examples

See asbh120 for an example of this block.

### See Also

FlightGear Preconfigured 6DoF Animation

Generate Run Script

Send net\_fdm Packet to FlightGear

Unpack net\_ctrl Packet from FlightGear

# Pilot Joystick

---

**Purpose** Provide joystick interface on Windows platform

**Library** Animation/Animation Support Utilities

**Description** The Pilot Joystick block provides a pilot joystick interface for a Windows platform. Roll, pitch, yaw, and throttle are mapped to the joystick *X*, *Y*, *R*, and *Z* channels respectively.

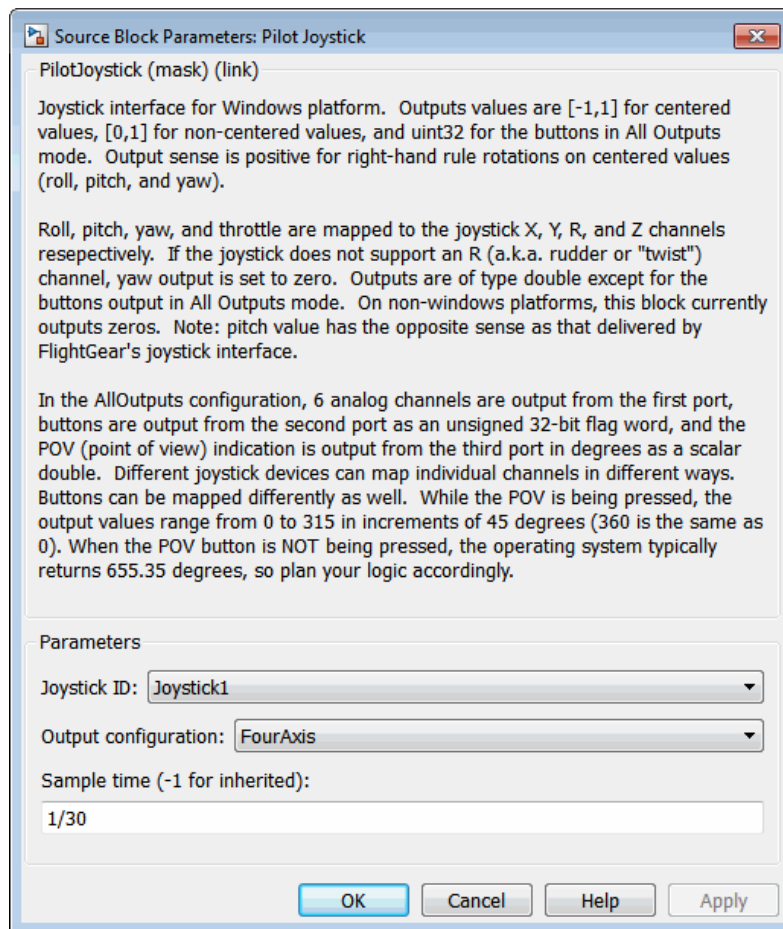


You can also configure the block to output all channels by setting the **Output configuration** parameter to AllOutputs.

This block does not produce deployable code.



## Dialog Box



### Joystick ID

Specify the joystick ID: Joystick 1, Joystick 2, or None.

### Output configuration

Specify the output configuration: FourAxis or AllOutputs (see Pilot Joystick All). FourAxis is the default.

# Pilot Joystick

---

## Sample time

Specify the sample time (-1 for inherited).

## Inputs and Outputs

The block has the following outputs.

### Four Axis Mode (All Double Precision Values)

Port Number	Output Range	Joystick	Description
1	[-1, 1]	[left, right]	Roll command
2	[-1, 1]	[forward/down, back/up]	Pitch command
3	[-1, 1]	[left, right]	Yaw command
4	[ 0, 1]	[min, max]	Throttle command

### All Outputs Mode (All Values Double Precision, Except for Buttons)

Port Number	Array Number	Channel	Output Range	Joystick	Description
1	1	X	[-1, 1]	[left, right]	Roll command
1	2	Y	[-1, 1]	[forward/down, back/up]	Pitch command
1	3	Z	[ 0, 1]	[min, max]	Throttle command
1	4	R	[-1, 1]	[left, right]	Yaw command
1	5	U	[ 0, 1]	[min, max]	U channel value
1	6	V	[ 0, 1]	[min, max]	V channel value

## All Outputs Mode (All Values Double Precision, Except for Buttons) (Continued)

Port Number	Array Number	Channel	Output Range	Joystick	Description
2		buttons			uint32 flagword containing up to 32 button states. Bit 0 is button 1, etc.
3		POV			Point-of-view hat value in degrees as a double. Zero degrees is straight ahead, 90 is to the left, etc.

Output values are [-1,1] for centered values, [0,1] for noncentered values, and uint32 for the buttons in All Outputs mode. Output sense is positive for right-hand rule rotations on centered values (roll, pitch, and yaw).

### Assumptions and Limitations

If the joystick does not support an *R* (rudder or “twist”) channel, yaw output is set to zero. Outputs are of type double except for the buttons output in AllOutputs mode, which is a uint32 flagword of bits. On non-Windows platforms, this block currently outputs zeros.

---

**Note** Pitch value has the opposite sense as that delivered by FlightGear’s joystick interface.

---

### See Also

Pilot Joystick All, Simulation Pace

# Pilot Joystick All

---

## Purpose

Provide joystick interface in All Outputs configuration on Windows platform

## Library

Animation/Animation Support Utilities

## Description

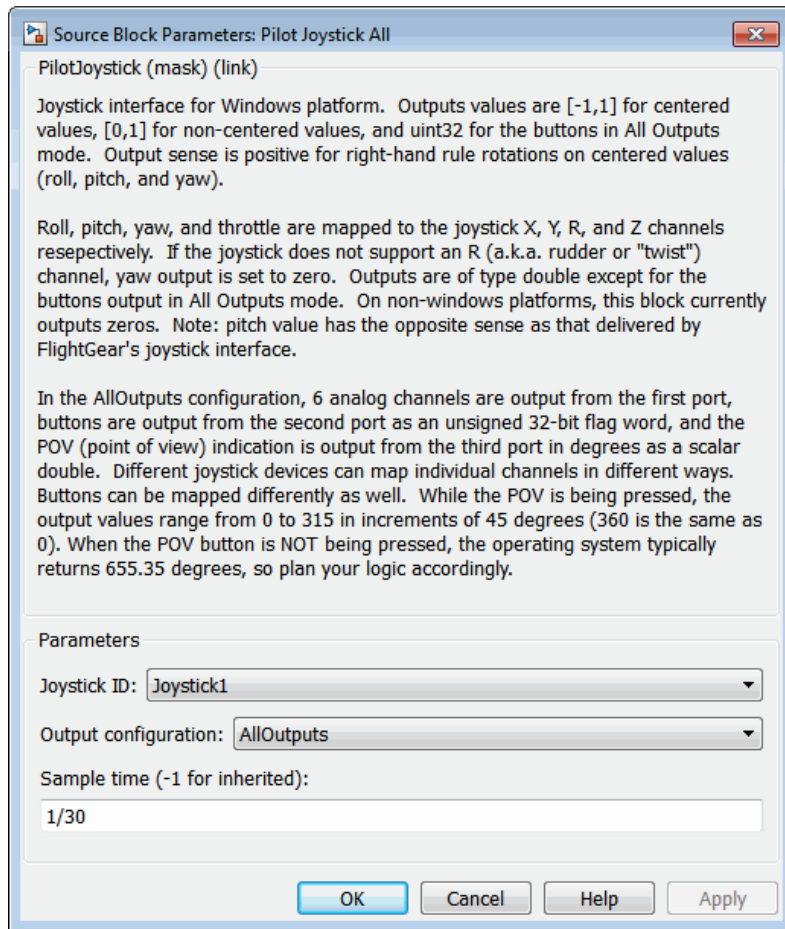


The Pilot Joystick All block provides a pilot joystick interface for a Windows platform. Analog is mapped to the joystick X, Y, Z, R, U, and V channels. Buttons and POV are mapped to up to 32 joystick button states and the joystick point-of-view hat.

You can also configure the block to output four axes by setting the **Output configuration** parameter to `FourAxis`.

This block does not produce deployable code.

## Dialog Box



### Joystick ID

Specify the joystick ID: Joystick 1, Joystick 2, or None.

### Output configuration

Specify the output configuration: FourAxis (see Pilot Joystick) or All1Outputs. All1Outputs is the default.

# Pilot Joystick All

---

## Sample time

Specify the sample time (-1 for inherited).

## Inputs and Outputs

The block has the following outputs.

### Four Axis Mode (All Double Precision Values)

Port Number	Output Range	Joystick	Description
1	[-1, 1]	[left, right]	Roll command
2	[-1, 1]	[forward/down, back/up]	Pitch command
3	[-1, 1]	[left, right]	Yaw command
4	[ 0, 1]	[min, max]	Throttle command

### All Outputs Mode (All Values Double Precision, Except for Buttons)

Port Number	Array Number	Channel	Output Range	Joystick	Description
1	1	X	[-1, 1]	[left, right]	Roll command
1	2	Y	[-1, 1]	[forward/down, back/up]	Pitch command
1	3	Z	[ 0, 1]	[min, max]	Throttle command
1	4	R	[-1, 1]	[left, right]	Yaw command
1	5	U	[ 0, 1]	[min, max]	U channel value
1	6	V	[ 0, 1]	[min, max]	V channel value

## All Outputs Mode (All Values Double Precision, Except for Buttons) (Continued)

Port Number	Array Number	Channel	Output Range	Joystick	Description
2		buttons			uint32 flagword containing up to 32 button states. Bit 0 is button 1, etc.
3		POV			Point-of-view hat value in degrees as a double. Zero degrees is straight ahead, 90 is to the left, etc.

Output values are [-1,1] for centered values, [0,1] for noncentered values, and uint32 for the buttons in All Outputs mode. Output sense is positive for right-hand rule rotations on centered values (roll, pitch, and yaw).

### Assumptions and Limitations

If the joystick does not support an R (rudder or “twist”) channel, yaw output is set to zero. Outputs are of type double except for the buttons output in AllOutputs mode, which is a uint32 flagword of bits. On non-Windows platforms, this block currently outputs zeros.

---

**Note** Pitch value has the opposite sense as that delivered by FlightGear’s joystick interface.

---

### See Also

Pilot Joystick, Simulation Pace

# Planetary Ephemeris

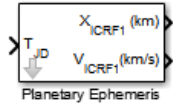
## Purpose

Implement position and velocity of astronomical objects

## Library

Environment/Celestial Phenomena

## Description



The Planetary Ephemeris block uses Chebyshev coefficients to implement the position and velocity of the target object relative to the specified center object for a given Julian date. The **Target** parameter specifies an astronomical object. The block implements the ephemerides using the **Center** parameter for an astronomical object as the reference.

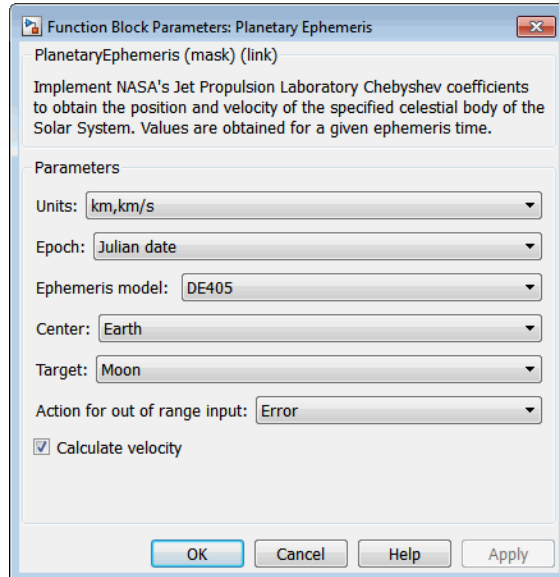
The block uses the Chebyshev coefficients that the NASA Jet Propulsion Laboratory provides.

---

**Tip** As input for the block, you can use the Aerospace Toolbox `juliandate` function to calculate the Julian date, or calculate your own Julian date, and input them using the Constant block.

---

## Dialog Box





## Units

Specify the output units:

<b>Units</b>	<b>Position</b>	<b>Velocity</b>
km, km/s	km	km/s
Au, AU/day	astronomical units (AU)	AU/day

## Epoch

Select one of the following:

- **Julian date**

Julian date to implement the position and velocity of the **Target** object. When this option is selected, the block has one input port.

- **$T_0$  and elapsed Julian time**

Julian date, specified by two block inputs:

- A fixed starting epoch ( $T_0$ ).
- Variable elapsed time between  $T_0$  and the desired model simulation time.

$T_0$  plus the variable elapsed time cannot exceed the maximum Julian date for the specified **Ephemerides**.

## Ephemeris model

Select one of the following ephemerides models defined by the Jet Propulsion Laboratory.

# Planetary Ephemeris

---

<b>Ephemerides Model</b>	<b>Description</b>
DE405	<p>Released in 1998. This ephemerides takes into account the Julian date range 2305424.50 (December 9, 1599 ) to 2525008.50 (February 20, 2201).</p> <p>This block implements these ephemerides with respect to the International Celestial Reference Frame version 1.0, adopted in 1998.</p>
DE421	<p>Released in 2008. This ephemerides takes into account the Julian date range 2414992.5 (December 4, 1899) to 2469808.5 (January 2, 2050).</p> <p>This block implements these ephemerides with respect to the International Celestial Reference Frame version 1.0, adopted in 1998.</p>
DE423	<p>Released in 2010. This ephemerides takes into account the Julian date range 2378480.5 (December 16, 1799) to 2524624.5 (February 1, 2200).</p> <p>This block implements these ephemerides with respect to the International Celestial Reference Frame version 2.0, adopted in 2010.</p>

## **Center**

Select a center body (astronomical object) or reference body as a point of reference for the **Target** barycenter position and velocity measurement.

## **Target**

Select a target body (astronomical object) or reference body as a point of reference for the barycenter position and velocity measurement.

## **Action for out of range input**

Specify the block behavior when the block inputs are out of range.

Action	Description
None	No action.
Warning	Warning in the MATLAB Command Window, model simulation continues.
Error (default)	MATLAB returns an exception, model simulation stops.

### Calculate velocity

Select this check box to calculate the velocity of the **Target** barycenter relative to the **Center** barycenter and add a second block output.

## Inputs and Outputs

Input	Dimension Type	Description
First	Scalar	<ul style="list-style-type: none"> <li>Julian date (<math>T_{JD}</math>) (default) — One input. Specify a date between the minimum and maximum Julian date.</li> <li>Fixed Julian date (<math>T0_{JD}</math>) plus the elapsed Julian time (<math>\Delta T_{JD}</math>) between the fixed date and the ephemeris time. — Two inputs, where the first input is <math>T0_{JD}</math> and the second input is <math>\Delta T_{JD}</math>. <math>\Delta T_{JD}</math> must be a positive number. The sum of <math>T0_{JD}</math> and <math>\Delta T_{JD}</math> must fall between the minimum and maximum Julian date.</li> </ul> <p>The block <b>Epoch</b> parameter controls the number of block inputs.</p>

# Planetary Ephemeris

Input	Dimension Type	Description
Second (Optional)	Scalar	<p>See the <b>Ephemerides</b> parameter for the minimum and maximum Julian dates.</p> <p><math>\Delta T_{JD}</math> — Elapsed Julian time (<math>\Delta T_{JD}</math>) between the fixed date and the ephemeris time. The sum of <math>T0_{JD}</math> and <math>\Delta T_{JD}</math> must fall between the minimum and maximum Julian date.</p> <p>See the <b>Ephemerides</b> parameter for the minimum and maximum Julian dates.</p>
Output	Dimension Type	Description
First	Vector	Barycenter position ( $X_{ICRF1}$ ) of the <b>Target</b> object relative to the barycenter of the <b>Center</b> object. Units are km or astronomical units (AU).
Second (Optional)	Vector	Velocity ( $V_{ICRF}$ ) of the barycenter of the <b>Target</b> object relative to the barycenter of the <b>Center</b> object. Units are km/s or astronomical units (AU)/day.

## Reference

- Folkner, W. M., J. G. Williams, D. H. Boggs, “The Planetary and Lunar Ephemeris DE 421,” *IPN Progress Report 42-178*, 2009.
- Ma, C. et al., “The International Celestial Reference Frame as Realized by Very Long Baseline Interferometry,” *Astronomical Journal*, Vol. 116, 516–546, 1998.
- Vallado, D. A., *Fundamentals of Astrodynamics and Applications*, McGraw-Hill, New York, 1997.

## See Also

Earth Nutation | Moon Libration |

## **External Web Sites**

- [http://ssd.jpl.nasa.gov/?planet\\_eph\\_export](http://ssd.jpl.nasa.gov/?planet_eph_export)

# Precision Pilot Model

**Purpose** Represent precision pilot model

**Library** Pilot Models

## Description



The Precision Pilot Model block represents the pilot model described in *Mathematical Models of Human Pilot Behavior*. (For more information, see [1]). This pilot model is a single input, single output (SISO) model that represents some aspects of human behavior when controlling aircraft. When modeling human pilot models, use this block for the most accuracy, compared to that provided by the Tustin Pilot Model and Crossover Pilot Model blocks.

This block is an extension of the Crossover Pilot Model block. When calculating the model, this block also takes into account the neuromuscular dynamics of the pilot. This block implements the following equation:

$$Y_p = K_p e^{-\tau s} \left( \frac{T_L s + 1}{T_I s + 1} \right) \left[ \frac{1}{(T_{N1} s + 1) \left( \frac{s^2}{\omega_N^2} + \frac{2\zeta_N}{\omega_N} s + 1 \right)} \right]$$

In this equation:

Variable	Description
$K_p$	Pilot gain.
$\tau$	Pilot delay time.
$T_L$	Time lead constant for the equalizer term.
$T_I$	Time lag constant.
$T_{N1}$	Time constant for the neuromuscular system.

Variable	Description
$\omega_N$	Undamped frequency for the neuromuscular system.
$\zeta_N$	Damping ratio for the neuromuscular system.

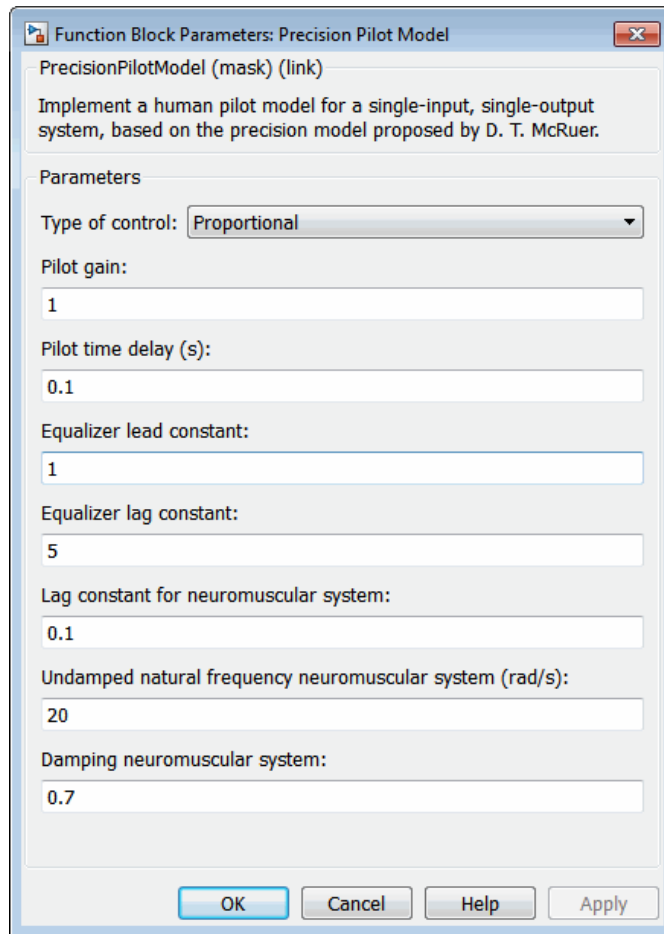
A sample value for the natural frequency and the damping ratio of a human is 20 rad/s and 0.7, respectively. The term containing the lead-lag term is the equalizer form. This form changes depending on the characteristics of the controlled system. A consistent behavior of the model can occur at different frequency ranges other than the crossover frequency.

This block has non-linear behavior. If you want to linearize the block (for example, with one of the Simulink `linmod` functions), you might need to change the Pade approximation order. The Precision Pilot Model block implementation incorporates the Simulink Transport Delay block with the **Pade order (for linearization)** parameter set to 2 by default. To change this value, use the `set_param` function, for example:

```
set_param(gcf, 'pade', '3')
```

# Precision Pilot Model

## Dialog Box



The image shows a software dialog box titled "Function Block Parameters: Precision Pilot Model". The dialog box contains the following elements:

- A title bar with a close button (X).
- A link: "PrecisionPilotModel (mask) (link)".
- A description: "Implement a human pilot model for a single-input, single-output system, based on the precision model proposed by D. T. McRuer."
- A "Parameters" section with the following fields:
  - "Type of control:" with a dropdown menu set to "Proportional".
  - "Pilot gain:" with a text input field containing "1".
  - "Pilot time delay (s):" with a text input field containing "0.1".
  - "Equalizer lead constant:" with a text input field containing "1".
  - "Equalizer lag constant:" with a text input field containing "5".
  - "Lag constant for neuromuscular system:" with a text input field containing "0.1".
  - "Undamped natural frequency neuromuscular system (rad/s):" with a text input field containing "20".
  - "Damping neuromuscular system:" with a text input field containing "0.7".
- Buttons at the bottom: "OK", "Cancel", "Help", and "Apply".

### Type of control

From the list, select one of the following options to specify the type of aircraft dynamics that you want to control. The equalizer form changes according to these values. For more information, see [2].



Option (Controlled Element Transfer Function)	Transfer Function of Controlled Element ( $Y_c$ )	Transfer Function of Pilot ( $Y_p$ )
Proportional	$K_c$	Lag-lead, $T_I \gg T_L$
Rate or velocity	$\frac{K_c}{s}$	1
Acceleration	$\frac{K_c}{s^2}$	Lead-lag, $T_L \gg T_I$
Second order	$\frac{K_c \omega_n^2}{s^2 + 2\zeta \omega_n s + \omega_n^2}$	Lead-lag if $\omega_m \ll 2/\tau$ . Lag-lead if $\omega_m \gg 2/\tau$ .

### Pilot gain

Specifies the pilot gain.

### Pilot time delay (s)

Specifies the total pilot time delay, in seconds. This value typically ranges from 0.1 s to 0.2 s.

### Equalizer lead constant

Specifies the equalizer lead constant.

### Equalizer lag constant

Specifies the equalizer lag constant.

### Lag constant for neuromuscular system

Specifies the neuromuscular system lag constant.

# Precision Pilot Model

---

## Undamped natural frequency neuromuscular system (rad/s)

Specifies the undamped natural frequency neuromuscular system in rad/s.

## Damping neuromuscular system

Specifies the damping neuromuscular system.

## Inputs and Outputs

Input	Dimension Type	Description
First	1-by-1	Contains the command for the signal that the pilot model controls.
Second	1-by-1	Contains the signal that the pilot model controls.

Output	Dimension Type	Description
First	1-by-1	Contains the command for the aircraft.

## References

[1] McRuer, D. T., Krendel, E., *Mathematical Models of Human Pilot Behavior*. Advisory Group on Aerospace Research and Development AGARDograph 180, Jan. 1974.

[2] McRuer, D. T., Graham, D., Krendel, E., and Reisener, W., *Human Pilot Dynamics in Compensatory Systems*. Air Force Flight Dynamics Lab. AFFDL-65-15. 1965.

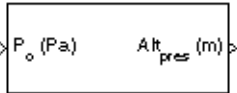
## See Also

Crossover Pilot Model | Tustin Pilot Model |

**Purpose** Calculate pressure altitude based on ambient pressure

**Library** Environment/Atmosphere

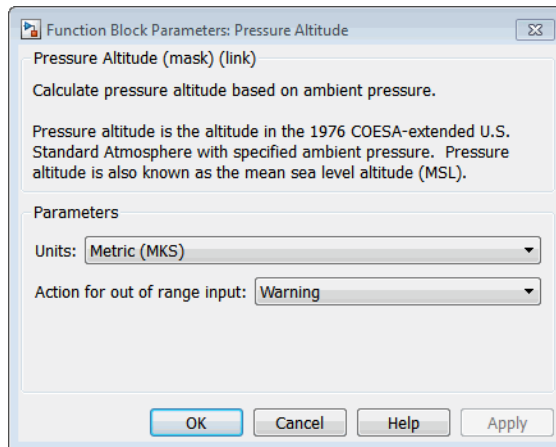
**Description** The Pressure Altitude block computes the pressure altitude based on ambient pressure. Pressure altitude is the altitude in the 1976 Committee on the Extension of the Standard Atmosphere (COESA) United States with specified ambient pressure.



Pressure altitude is also known as the mean sea level (MSL) altitude.

The Pressure Altitude block icon displays the input and output units selected from the **Units** list.

## Dialog Box



**Units** Specifies the input units:

<b>Units</b>	<b>Pstatic</b>	<b>Alt_p</b>
Metric (MKS)	Pascal	Meters
English	Pound force per square inch	Feet

# Pressure Altitude

---

## Action for out of range input

Specify if out-of-range input invokes a warning, error, or no action.

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the static pressure.

Output	Dimension Type	Description
First		Contains the pressure altitude.

## Assumptions and Limitations

Below the pressure of 0.3961 Pa (approximately 0.00006 psi) and above the pressure of 101325 Pa (approximately 14.7 psi), altitude values are extrapolated logarithmically.

Air is assumed to be dry and an ideal gas.

## Reference

U.S. Standard Atmosphere, 1976, U.S. Government Printing Office, Washington, D.C.

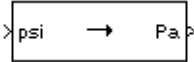
## See Also

COESA Atmosphere Model

**Purpose** Convert from pressure units to desired pressure units

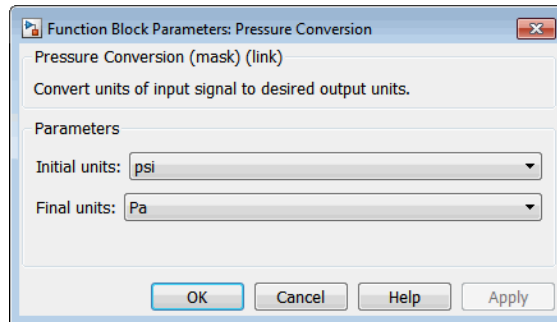
**Library** Utilities/Unit Conversions

**Description** The Pressure Conversion block computes the conversion factor from specified input pressure units to specified output pressure units and applies the conversion factor to the input signal.



The Pressure Conversion block icon displays the input and output units selected from the **Initial units** and the **Final units** lists.

## Dialog Box



**Initial units**  
Specifies the input units.

**Final units**  
Specifies the output units.

The following conversion units are available:

psi	Pound mass per square inch
Pa	Pascals
psf	Pound mass per square foot
atm	Atmospheres

# Pressure Conversion

---

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the pressure in initial pressure units.

Output	Dimension Type	Description
First		Contains the pressure in final pressure units.

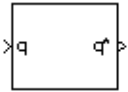
## See Also

Acceleration Conversion  
Angle Conversion  
Angular Acceleration Conversion  
Angular Velocity Conversion  
Density Conversion  
Force Conversion  
Length Conversion  
Mass Conversion  
Temperature Conversion  
Velocity Conversion

**Purpose** Calculate conjugate of quaternion

**Library** Utilities/Math Operations

**Description** The Quaternion Conjugate block calculates the conjugate for a given quaternion.



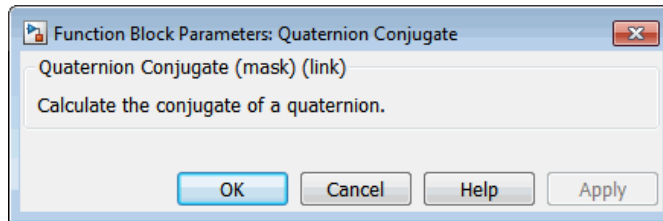
The quaternion has the form of

$$q = q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3$$

The quaternion conjugate has the form of

$$q' = q_0 - \mathbf{i}q_1 - \mathbf{j}q_2 - \mathbf{k}q_3$$

## Dialog Box



## Inputs and Outputs

Input	Dimension Type	Description
First	Quaternion or vector	Contains quaternions in the form of $[q_0, r_0, \dots, q_1, r_1, \dots, q_2, r_2, \dots, q_3, r_3, \dots]$ .

Output	Dimension Type	Description
First	Quaternion conjugate or vector	Contains quaternion conjugates in the form of $[q_0', r_0', \dots, q_1', r_1', \dots, q_2', r_2', \dots, q_3', r_3', \dots]$ .

**See Also** Quaternion Division

# Quaternion Conjugate

---

Quaternion Inverse

Quaternion Modulus

Quaternion Multiplication

Quaternion Norm

Quaternion Normalize

Quaternion Rotation



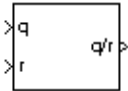
## Purpose

Divide quaternion by another quaternion

## Library

Utilities/Math Operations

## Description



The Quaternion Division block divides a given quaternion by another.

The quaternions have the form of

$$q = q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3$$

and

$$r = r_0 + \mathbf{i}r_1 + \mathbf{j}r_2 + \mathbf{k}r_3$$

The resulting quaternion from the division has the form of

$$t = \frac{q}{r} = t_0 + \mathbf{i}t_1 + \mathbf{j}t_2 + \mathbf{k}t_3$$

where

$$t_0 = \frac{(r_0q_0 + r_1q_1 + r_2q_2 + r_3q_3)}{r_0^2 + r_1^2 + r_2^2 + r_3^2}$$

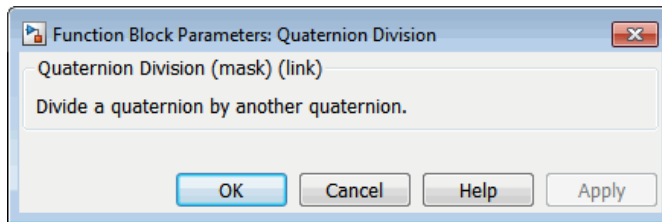
$$t_1 = \frac{(r_0q_1 - r_1q_0 - r_2q_3 + r_3q_2)}{r_0^2 + r_1^2 + r_2^2 + r_3^2}$$

$$t_2 = \frac{(r_0q_2 + r_1q_3 - r_2q_0 - r_3q_1)}{r_0^2 + r_1^2 + r_2^2 + r_3^2}$$

$$t_3 = \frac{(r_0q_3 - r_1q_2 + r_2q_1 - r_3q_0)}{r_0^2 + r_1^2 + r_2^2 + r_3^2}$$

# Quaternion Division

## Dialog Box



## Inputs and Outputs

Input	Dimension Type	Description
First	Quaternion or vector	Contains quaternions in the form of $[q_0, p_0, \dots, q_1, p_1, \dots, q_2, p_2, \dots, q_3, p_3, \dots]$ .
Second	Quaternion or vector	Contains quaternions in the form of $[s_0, r_0, \dots, s_1, r_1, \dots, s_2, r_2, \dots, s_3, r_3, \dots]$ .

Output	Dimension Type	Description
First	Quaternion or vector	Contains resulting quaternion or vector of resulting quaternions from division.

The output is the resulting quaternion from the division or vector of resulting quaternions from division.

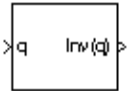
## See Also

- Quaternion Conjugate
- Quaternion Inverse
- Quaternion Modulus
- Quaternion Multiplication
- Quaternion Norm
- Quaternion Normalize
- Quaternion Rotation

**Purpose** Calculate inverse of quaternion

**Library** Utilities/Math Operations

**Description** The Quaternion Inverse block calculates the inverse for a given quaternion.



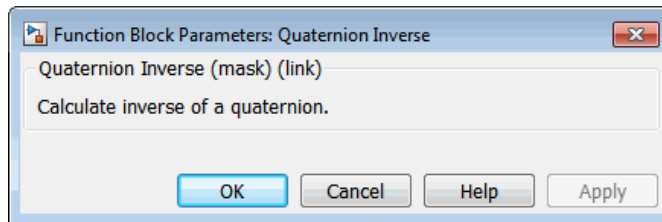
The quaternion has the form of

$$q = q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3$$

The quaternion inverse has the form of

$$q^{-1} = \frac{q_0 - \mathbf{i}q_1 - \mathbf{j}q_2 - \mathbf{k}q_3}{q_0^2 + q_1^2 + q_2^2 + q_3^2}$$

## Dialog Box



## Inputs and Outputs

Input	Dimension Type	Description
First	Quaternion or vector	Contains quaternions in the form of $[q_0, r_0, \dots, q_1, r_1, \dots, q_2, r_2, \dots, q_3, r_3, \dots]$ .
Output	Dimension Type	Description
First	Quaternion inverse or vector	Contains quaternion inverse or vector of quaternion inverses.

# Quaternion Inverse

---

## See Also

[Quaternion Conjugate](#)

[Quaternion Division](#)

[Quaternion Modulus](#)

[Quaternion Multiplication](#)

[Quaternion Norm](#)

[Quaternion Normalize](#)

[Quaternion Rotation](#)

**Purpose** Calculate modulus of quaternion

**Library** Utilities/Math Operations

**Description** The Quaternion Modulus block calculates the magnitude for a given quaternion.



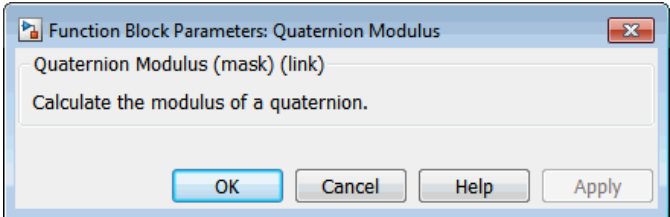
The quaternion has the form of

$$q = q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3$$

The quaternion modulus has the form of

$$|q| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}$$

**Dialog Box**



**Inputs and Outputs**

Input	Dimension Type	Description
First	Quaternion or vector	Contains quaternions in the form of $[q_0, r_0, \dots, q_1, r_1, \dots, q_2, r_2, \dots, q_3, r_3, \dots]$ .
Output	Dimension Type	Description
First	Quaternion modulus or vector	Contains quaternion modulus or vector of quaternion modulus in the form of $[ q ,  r , \dots]$ .

# Quaternion Modulus

---

## **See Also**

[Quaternion Conjugate](#)

[Quaternion Division](#)

[Quaternion Inverse](#)

[Quaternion Multiplication](#)

[Quaternion Norm](#)

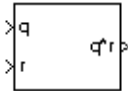
[Quaternion Normalize](#)

[Quaternion Rotation](#)

**Purpose** Calculate product of two quaternions

**Library** Utilities/Math Operations

**Description** The Quaternion Multiplication block calculates the product for two given quaternions.



The quaternions have the form of

$$q = q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3$$

and

$$r = r_0 + \mathbf{i}r_1 + \mathbf{j}r_2 + \mathbf{k}r_3$$

The quaternion product has the form of

$$t = q \times r = t_0 + \mathbf{i}t_1 + \mathbf{j}t_2 + \mathbf{k}t_3$$

where

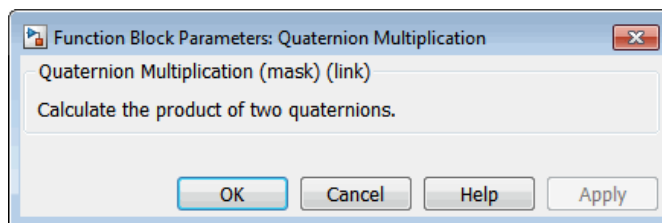
$$t_0 = (r_0q_0 - r_1q_1 - r_2q_2 - r_3q_3)$$

$$t_1 = (r_0q_1 + r_1q_0 - r_2q_3 + r_3q_2)$$

$$t_2 = (r_0q_2 + r_1q_3 + r_2q_0 - r_3q_1)$$

$$t_3 = (r_0q_3 - r_1q_2 + r_2q_1 + r_3q_0)$$

**Dialog Box**



# Quaternion Multiplication

---

## Inputs and Outputs

Input	Dimension Type	Description
First	Quaternion or vector	Contains quaternions in the form of $[q_0, p_0, \dots, q_1, p_1, \dots, q_2, p_2, \dots, q_3, p_3, \dots]$ .
Second	Quaternion or vector	Contains quaternions in the form of $[s_0, r_0, \dots, s_1, r_1, \dots, s_2, r_2, \dots, s_3, r_3, \dots]$ .

Output	Dimension Type	Description
First	Quaternion product or vector	Contains quaternion product or vector of quaternion products.

## See Also

Quaternion Conjugate

Quaternion Division

Quaternion Inverse

Quaternion Modulus

Quaternion Norm

Quaternion Normalize

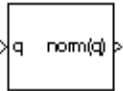
Quaternion Rotation



**Purpose** Calculate norm of quaternion

**Library** Utilities/Math Operations

**Description** The Quaternion Norm block calculates the norm for a given quaternion. The quaternion has the form of

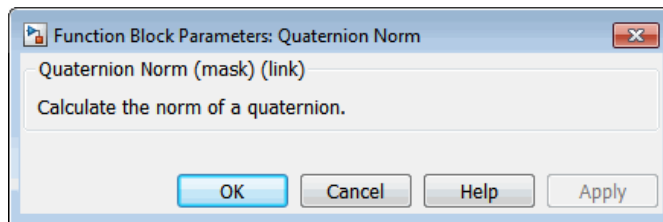


$$q = q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3$$

The quaternion norm has the form of

$$\text{norm}(q) = q_0^2 + q_1^2 + q_2^2 + q_3^2$$

## Dialog Box



## Inputs and Outputs

Input	Dimension Type	Description
First	Quaternion or vector	Contains quaternions in the form of $[q_0, r_0, \dots, q_1, r_1, \dots, q_2, r_2, \dots, q_3, r_3, \dots]$ .

Output	Dimension Type	Description
First	Quaternion norm or vector	Contains quaternion norm or vector of quaternion norms in the form of $[\text{norm}(q), \text{norm}(r), \dots]$ .

**See Also** Quaternion Conjugate

# Quaternion Norm

---

Quaternion Division

Quaternion Inverse

Quaternion Modulus

Quaternion Multiplication

Quaternion Normalize

Quaternion Rotation

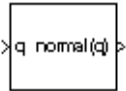
## Purpose

Normalize quaternion

## Library

Utilities/Math Operations

## Description



The Quaternion Normalize block calculates a normalized quaternion for a given quaternion.

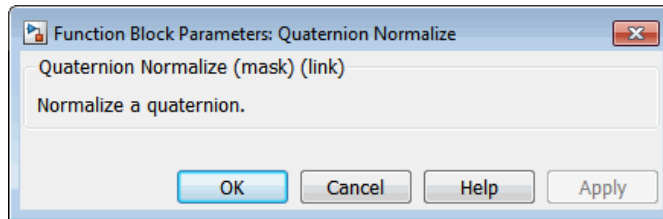
The quaternion has the form of

$$q = q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3$$

The normalized quaternion has the form of

$$\text{normal}(q) = \frac{q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3}{\sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}}$$

## Dialog Box



## Inputs and Outputs

Input	Dimension Type	Description
First	Quaternion or vector	Contains quaternions in the form of $[q_0, r_0, \dots, q_1, r_1, \dots, q_2, r_2, \dots, q_3, r_3, \dots]$ .
Output	Dimension Type	Description
First	Normalized quaternion or vector	Contains normalized quaternion or vector of normalized quaternions.

# Quaternion Normalize

---

## See Also

[Quaternion Conjugate](#)

[Quaternion Division](#)

[Quaternion Inverse](#)

[Quaternion Modulus](#)

[Quaternion Multiplication](#)

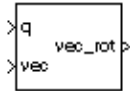
[Quaternion Norm](#)

[Quaternion Rotation](#)

**Purpose** Rotate vector by quaternion

**Library** Utilities/Math Operations

**Description** The Quaternion Rotation block rotates a vector by a quaternion.



The quaternion has the form of

$$q = q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3$$

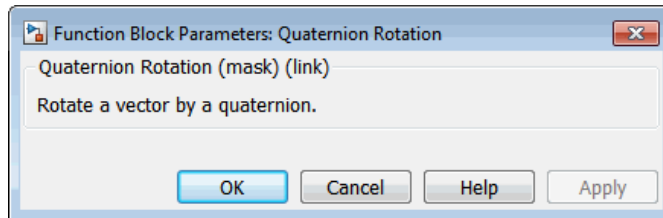
The vector has the form of

$$v = \mathbf{i}v_1 + \mathbf{j}v_2 + \mathbf{k}v_3$$

The rotated vector has the form of

$$v' = \begin{bmatrix} v_1' \\ v_2' \\ v_3' \end{bmatrix} = \begin{bmatrix} (1 - 2q_2^2 - 2q_3^2) & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & (1 - 2q_1^2 - 2q_3^2) & 2(q_2q_3 + q_0q_1) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & (1 - 2q_1^2 - 2q_2^2) \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

## Dialog Box



# Quaternion Rotation

---

## Inputs and Outputs

Input	Dimension Type	Description
First	Quaternion or vector	Contains quaternions in the form of $[q_0, r_0, \dots, q_1, r_1, \dots, q_2, r_2, \dots, q_3, r_3, \dots]$ .
Second	Vector	Contains vector or vector of vectors in the form of $[v_1, u_1, \dots, v_2, u_2, \dots, v_3, u_3, \dots]$ .

Output	Dimension Type	Description
First	Rotated quaternion or vector	Contains rotated vector or vector of rotated vectors.

## See Also

- Quaternion Conjugate
- Quaternion Division
- Quaternion Inverse
- Quaternion Modulus
- Quaternion Multiplication
- Quaternion Norm
- Quaternion Normalize

# Quaternions to Direction Cosine Matrix

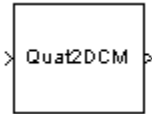
## Purpose

Convert quaternion vector to direction cosine matrix

## Library

Utilities/Axes Transformations

## Description



The Quaternions to Direction Cosine Matrix block transforms the four-element unit quaternion vector  $(q_0, q_1, q_2, q_3)$  into a 3-by-3 direction cosine matrix (DCM). The outputted DCM performs the coordinate transformation of a vector in inertial axes to a vector in body axes.

Using quaternion algebra, if a point  $P$  is subject to the rotation described by a quaternion  $q$ , it changes to  $P'$  given by the following relationship:

$$\begin{aligned}P' &= qPq^c \\q &= q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3 \\q^c &= q_0 - \mathbf{i}q_1 - \mathbf{j}q_2 - \mathbf{k}q_3 \\P &= 0 + \mathbf{i}x + \mathbf{j}y + \mathbf{k}z\end{aligned}$$

Expanding  $P'$  and collecting terms in  $x$ ,  $y$ , and  $z$  gives the following for  $P'$  in terms of  $P$  in the vector quaternion format:

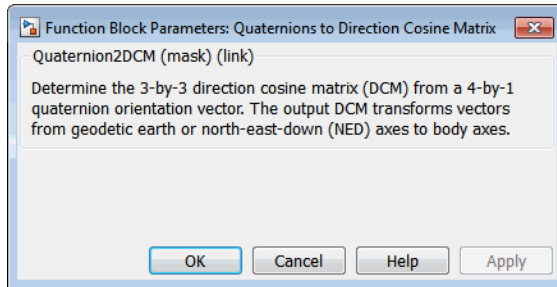
$$P' = \begin{bmatrix} 0 \\ x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 0 \\ (q_0^2 + q_1^2 - q_2^2 - q_3^2)x + 2(q_1q_2 - q_0q_3)y + 2(q_1q_3 + q_0q_2)z \\ 2(q_0q_3 + q_1q_2)x + (q_0^2 - q_1^2 + q_2^2 - q_3^2)y + 2(q_2q_3 - q_0q_1)z \\ 2(q_1q_3 - q_0q_2)x + 2(q_0q_1 + q_2q_3)y + (q_0^2 - q_1^2 - q_2^2 + q_3^2)z \end{bmatrix}$$

Since individual terms in  $P'$  are linear combinations of terms in  $x$ ,  $y$ , and  $z$ , a matrix relationship to rotate the vector  $(x, y, z)$  to  $(x', y', z')$  can be extracted from the preceding. This matrix rotates a vector in inertial axes, and hence is transposed to generate the DCM that performs the coordinate transformation of a vector in inertial axes into body axes.

# Quaternions to Direction Cosine Matrix

$$DCM = \begin{bmatrix} (q_0^2 + q_1^2 - q_2^2 - q_3^2) & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & (q_0^2 - q_1^2 + q_2^2 - q_3^2) & 2(q_2q_3 + q_0q_1) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & (q_0^2 - q_1^2 - q_2^2 + q_3^2) \end{bmatrix}$$

## Dialog Box



## Inputs and Outputs

Input	Dimension Type	Description
First	4-by-1 quaternion vector	Contains the quaternion vector.

Output	Dimension Type	Description
First	3-by-3 direction cosine matrix.	Contains the direction cosine matrix.

## See Also

[Direction Cosine Matrix to Rotation Angles](#)  
[Direction Cosine Matrix to Quaternions](#)  
[Rotation Angles to Direction Cosine Matrix](#)  
[Rotation Angles to Quaternions](#)

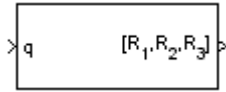


# Quaternions to Rotation Angles

**Purpose** Determine rotation vector from quaternion

**Library** Utilities/Axes Transformations

## Description



The Quaternions to Rotation Angles block converts the four-element quaternion vector  $(q_0, q_1, q_2, q_3)$  into the rotation described by the three rotation angles (R1, R2, R3). The block generates the conversion by comparing elements in the direction cosine matrix (DCM) as a function of the rotation angles. The elements in the DCM are functions of a unit quaternion vector. For example, for the rotation order z-y-x, the DCM is defined as:

$$DCM = \begin{bmatrix} \cos \theta \cos \psi & \cos \theta \sin \psi & -\sin \theta \\ (\sin \phi \sin \theta \cos \psi - \cos \phi \sin \psi) & (\sin \phi \sin \theta \sin \psi + \cos \phi \cos \psi) & \sin \phi \cos \theta \\ (\cos \phi \sin \theta \cos \psi + \sin \phi \sin \psi) & (\cos \phi \sin \theta \sin \psi - \sin \phi \cos \psi) & \cos \phi \cos \theta \end{bmatrix}$$

The DCM defined by a unit quaternion vector is:

$$DCM = \begin{bmatrix} (q_0^2 + q_1^2 - q_2^2 - q_3^2) & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & (q_0^2 - q_1^2 + q_2^2 - q_3^2) & 2(q_2q_3 + q_0q_1) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & (q_0^2 - q_1^2 - q_2^2 + q_3^2) \end{bmatrix}$$

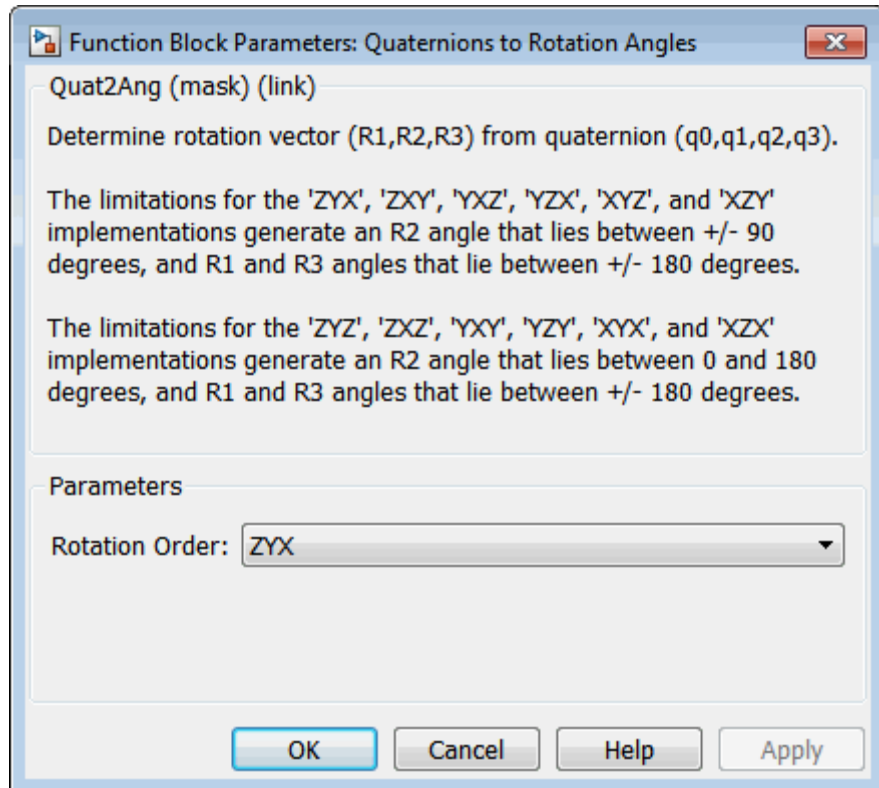
From the preceding equation, you can derive the following relationships between DCM elements and individual rotation angles for a ZYX rotation order:

$$\begin{aligned} \phi &= \text{atan}(DCM(2,3), DCM(3,3)) \\ &= \text{atan}(2(q_2q_3 + q_0q_1), (q_0^2 - q_1^2 - q_2^2 + q_3^2)) \\ \theta &= \text{asin}(-DCM(1,3)) \\ &= \text{asin}(-2(q_1q_3 - q_0q_2)) \\ \psi &= \text{atan}(DCM(1,2), DCM(1,1)) \\ &= \text{atan}(2(q_1q_2 + q_0q_3), (q_0^2 + q_1^2 - q_2^2 - q_3^2)) \end{aligned}$$

# Quaternions to Rotation Angles

where  $\Psi$  is R1,  $\Theta$  is R2, and  $\Phi$  is R3.

## Dialog Box



### Rotation Order

Specifies the output rotation order for three rotation angles. From the list, select ZYX, ZYZ, ZXY, ZXZ, YXZ, YXY, YZX, YZY, XYZ, XYX, XZY, or XZX. The default is ZYX.

## Inputs and Outputs

Input	Dimension Type	Description
First	4-by-1 quaternion vector	Contains the quaternion vector.

Output	Dimension Type	Description
First	3-by-3 vector	Contains the rotation angles, in radians.

## Assumptions and Limitations

The limitations for the 'ZYX', 'ZXY', 'YXZ', 'YZX', 'XYZ', and 'XZY' implementations generate an R2 angle that is between  $\pm 90$  degrees, and R1 and R3 angles that are between  $\pm 180$  degrees.

The limitations for the 'YZY', 'ZXZ', 'YXY', 'YZY', 'YXY', and 'XZX' implementations generate an R2 angle that is between 0 and 180 degrees, and R1 and R3 angles that are between  $\pm 180$  degrees.

## See Also

[Direction Cosine Matrix to Rotation Angles](#)

[Direction Cosine Matrix to Quaternions](#)

[Quaternions to Direction Cosine Matrix](#)

[Rotation Angles to Direction Cosine Matrix](#)

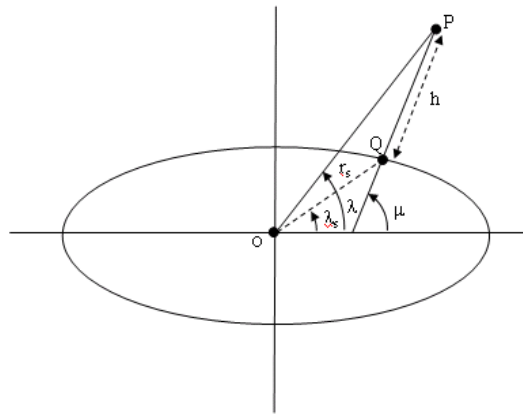
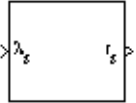
[Rotation Angles to Quaternions](#)

# Radius at Geocentric Latitude

**Purpose** Estimate radius of ellipsoid planet at geocentric latitude

**Library** Flight Parameters

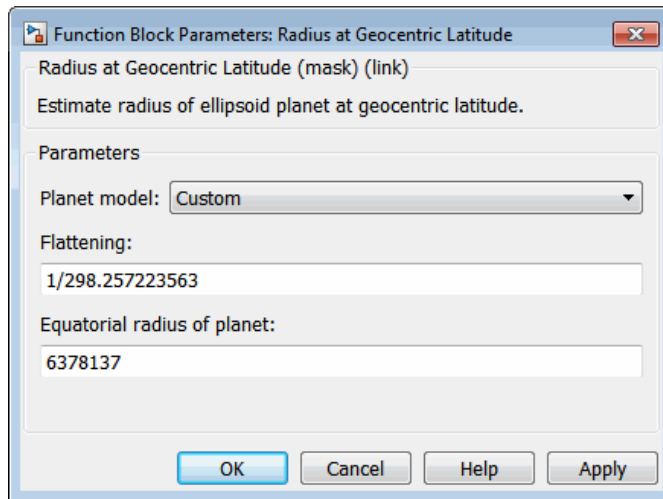
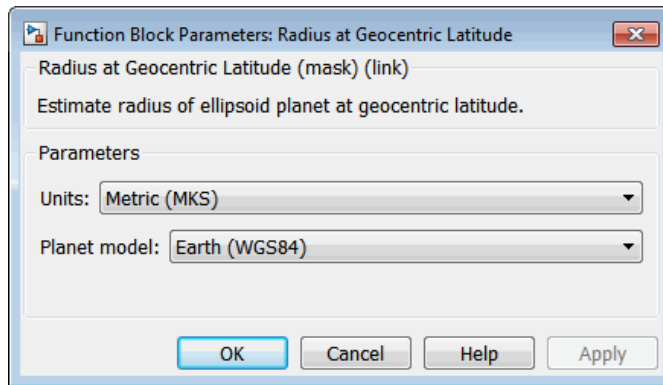
**Description** The Radius at Geocentric Latitude block estimates the radius ( $r_s$ ) of an ellipsoid planet at a particular geocentric latitude ( $\lambda_g$ ).



The following equation estimates the ellipsoid radius ( $r_s$ ) using flattening ( $\bar{f}$ ), geocentric latitude ( $\bar{\lambda}_g$ ), and equatorial radius ( $\bar{R}$ ).

$$r_s = \sqrt{\frac{\bar{R}^2}{1 + \left[ \frac{1}{(1 - \bar{f})^2} - 1 \right] \sin^2 \bar{\lambda}_g}}$$

## Dialog Box



### Units

Specifies the parameter and output units:

# Radius at Geocentric Latitude

---

<b>Units</b>	<b>Equatorial Radius</b>	<b>Radius at Geocentric Latitude</b>
Metric (MKS)	Meters	Meters
English	Feet	Feet

This option is only available when **Planet model** is set to Earth (WGS84).

## **Planet model**

Specifies the planet model to use:

Custom

Earth (WGS84)

## **Flattening**

Specifies the flattening of the planet. This option is only available with **Planet model** set to Custom.

## **Equatorial radius of planet**

Specifies the radius of the planet at its equator. This option is only available with **Planet model** set to Custom.

## **Inputs and Outputs**

<b>Input</b>	<b>Dimension Type</b>	<b>Description</b>
First		Contains the geocentric latitude, in degrees.

<b>Output</b>	<b>Dimension Type</b>	<b>Description</b>
First		Contains the radius of planet at geocentric latitude, in the same as the units as flattening.

## References

Stevens, B. L., and F. L. Lewis, *Aircraft Control and Simulation*, John Wiley & Sons, New York, 1992.

Zipfel, P. H., *Modeling and Simulation of Aerospace Vehicle Dynamics*, AIAA Education Series, Reston, Virginia, 2000.

## See Also

ECEF Position to LLA

Direction Cosine Matrix ECEF to NED

Direction Cosine Matrix ECEF to NED to Latitude and Longitude

Geocentric to Geodetic Latitude

Geodetic to Geocentric Latitude

LLA to ECEF Position

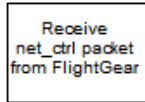
# Receive net\_ctrl Packet from FlightGear

---

**Purpose** Receive net\_ctrl packet from FlightGear

**Library** Animation/Flight Simulator Interfaces

## Description



The Receive net\_ctrl Packet from FlightGear block receives a network control and environment data packet, `net_ctrl`, from the simulation of a Simulink model in the FlightGear simulator, or from a FlightGear session. This data packet is compatible with a particular version of FlightGear flight simulator. All the signals supported by the FlightGear `net_ctrl` data packet are supported by this block. The signals are arranged into multiple groups. Any group can be turned on or off. Zeros are inserted for packet values that are part of inactive signal groups.

In Rapid Accelerator mode, if you run a model that contains this block, the block produces zeroes (0s). It does not produce deployable code. In Accelerator mode, the block works as expected.

For details on signals and signal groups, see “Inputs and Outputs” on page 4-416.

Supported FlightGear versions are:

- v0.9.3
- v0.9.8/0.9.8a
- v0.9.9
- v0.9.10
- v1.0
- v1.9.1
- v2.0
- v2.4
- v2.6
- v2.8



## Determining the Source IP Address

To determine the source IP address, you can use one of several techniques, such as:

- Use 127.0.0.1 for the local computer (localhost).
- Ping another computer from a Windows cmd.exe (or UNIX shell) prompt:

```
C:\> ping andyspc
```

```
Pinging andyspc [144.213.175.92] with 32 bytes of data:
```

```
Reply from 144.213.175.92: bytes=32 time=30ms TTL=253
Reply from 144.213.175.92: bytes=32 time=20ms TTL=253
Reply from 144.213.175.92: bytes=32 time=20ms TTL=253
Reply from 144.213.175.92: bytes=32 time=20ms TTL=253
```

```
Ping statistics for 144.213.175.92:
```

```
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 20ms, Maximum = 30ms, Average = 22ms
```

- On a Windows machine, type ipconfig and use the returned IP address:

```
H:\>ipconfig
```

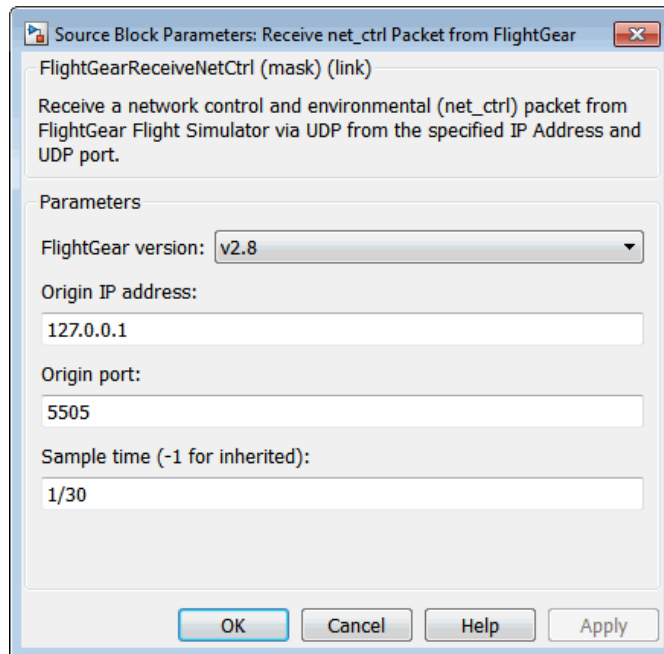
```
Windows IP Configuration
```

```
Ethernet adapter Local Area Connection:
```

```
    Connection-specific DNS Suffix . . :
    IP Address. . . . . : 192.168.42.178
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.42.254
```

# Receive net\_ctrl Packet from FlightGear

## Dialog Box



### FlightGear version

Select your FlightGear software version.

Supported FlightGear versions are: v0.9.3, v0.9.8/0.9.8a, v0.9.9, v0.9.10, v1.0, v1.9.1, v2.0, v2.4, v2.6, v2.8.

### Origin IP address

Enter a valid IP address as a dot-decimal string. This IP address must be the address of the PC from which FlightGear is run.

For example, 10.10.10.3. You can also use a MATLAB expression that returns a valid IP address as a string. If FlightGear is run on the local PC, leave the default value of 127.0.0.1 (localhost).

# Receive net\_ctrl Packet from FlightGear

## Origin port

UDP port that the block accepts data from. The sender sends data to the port specified in this parameter. This value must match the **Origin port** parameter of the Generate Run Script block. It must be a unique port number that no other application on the PC uses. The site, [http://en.wikipedia.org/wiki/List\\_of\\_TCP\\_and\\_UDP\\_port\\_numbers](http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers), lists commonly known UDP port numbers. To identify UDP port numbers already in use on your PC, type:

```
netstat -a -p UDP
```

## Sample time

Specify the sample time (-1 for inherited).

## Inputs and Outputs

Output	Dimension	Type	Description
First	Platform	Type	Contains the controls information from FlightGear in uint8 format.
	Windows and Linux	744-by-1 vector (for FlightGear v2.8, v2.6, v2.4, v2.0, v1.9, v1.0, v0.9.10)	
		536-by-1 vector (for FlightGear v0.9.9)	
		392-by-1 vector (for FlightGear v0.9.8)	
		424-by-1 vector (for FlightGear v0.9.3)	

# Receive net\_ctrl Packet from FlightGear

---

Output	Dimension	Type	Description
Mac OS	744-by-1	vector	(for FlightGear v2.6*, v2.8)
	732-by-1	vector	(for FlightGear v2.0, v2.4, v1.9, v1.0, v0.9.10)
	524-by-1	vector	(for FlightGear v0.9.9)
	504-by-1	vector	(for FlightGear v0.9.8)
	400-by-1	vector	(for FlightGear v0.9.3)

\* On a Macintosh system with FlightGear 2.6, you might see unexpected results (for example, very large or very small data values). For more information, see “Macintosh Platform and FlightGear 2.6” on page 2-51.

## Examples

See the asbh120 example.

## See Also

FlightGear Preconfigured 6DoF Animation

Generate Run Script

Send net\_fdm Packet to FlightGear

Unpack net\_ctrl Packet from FlightGear

## Purpose

Calculate relative atmospheric ratios

## Library

Flight Parameters

## Description

> Mach	$\theta$
> $\gamma$	$\sqrt{\theta}$
> $T_o$ (K)	
> $P_o$ (Pa)	$\delta$
> $\rho_o$ (kg/m <sup>3</sup> )	$\sigma$

The Relative Ratio block computes the relative atmospheric ratios, including relative temperature ratio ( $\theta$ ),  $\sqrt{\theta}$ , relative pressure ratio ( $\delta$ ), and relative density ratio ( $\sigma$ ).

$\theta$  represents the ratio of the air stream temperature at a chosen reference station relative to sea level standard atmospheric conditions.

$$\theta = \frac{T}{T_0}$$

$\delta$  represents the ratio of the air stream pressure at a chosen reference station relative to sea level standard atmospheric conditions.

$$\delta = \frac{P}{P_0}$$

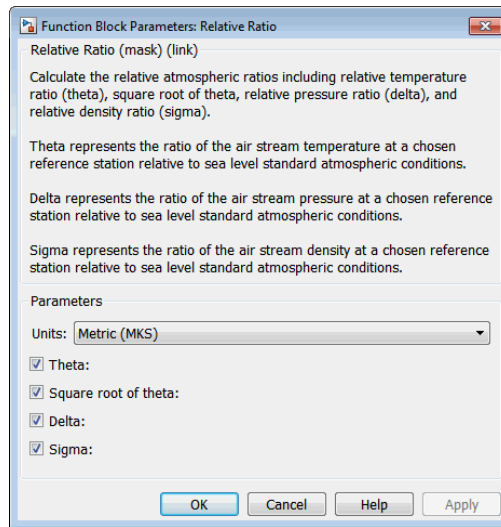
$\sigma$  represents the ratio of the air stream density at a chosen reference station relative to sea level standard atmospheric conditions.

$$\sigma = \frac{\rho}{\rho_0}$$

The Relative Ratio block icon displays the input units selected from the **Units** list.

# Relative Ratio

## Dialog Box



## Units

Specifies the input units:

<b>Units</b>	<b>Tstatic</b>	<b>Pstatic</b>	<b>rho_static</b>
Metric (MKS)	Kelvin	Pascal	Kilograms per cubic meter
English	Degrees Rankine	Pound force per square inch	Slug per cubic foot

## Theta

When selected, the  $\theta$  is calculated and static temperature is a required input.

## Square root of theta

When selected, the  $\sqrt{\theta}$  is calculated and static temperature is a required input.

## Delta

When selected, the  $\delta$  is calculated and static pressure is a required input.

## Sigma

When selected, the  $\sigma$  is calculated and static density is a required input.

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the Mach number.
Second		Contains the static temperature.
Third		Contains the static pressure.
Fourth		Contains the static density.

Output	Dimension Type	Description
First		Contains the $\theta$ .
Second		Contains the $\sqrt{\theta}$ .
Third		Contains the $\delta$ .
Fourth		Contains the $\sigma$ .

## Assumptions

For cases in which total temperature, total pressure, or total density ratio is desired (Mach number is nonzero), the total temperature, total pressure, and total densities are calculated assuming perfect gas (with constant molecular weight, constant pressure specific heat, and constant specific heat ratio) and dry air.

## Reference

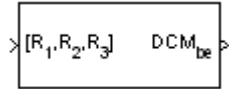
*Aeronautical Vestpocket Handbook*, United Technologies Pratt & Whitney, August, 1986.

# Rotation Angles to Direction Cosine Matrix

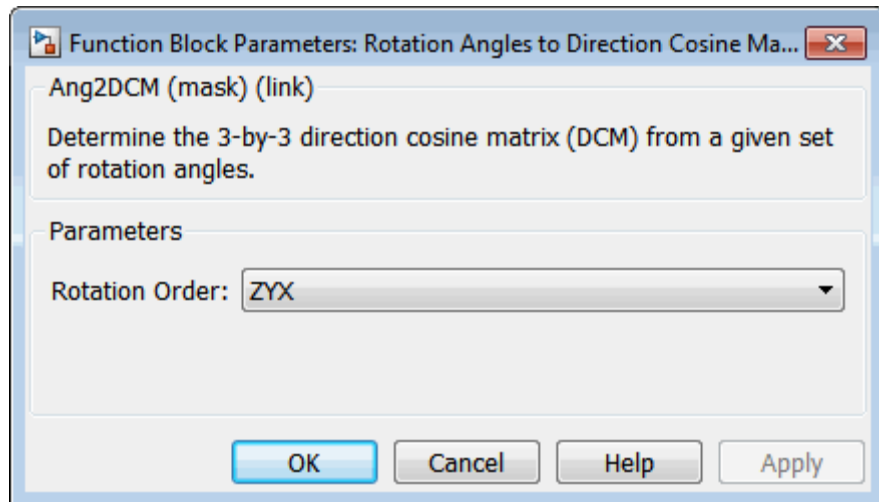
**Purpose** Convert rotation angles to direction cosine matrix

**Library** Utilities/Axes Transformations

**Description** The Rotation Angles to Direction Cosine Matrix block determines the direction cosine matrix (DCM) from a given set of rotation angles, R1, R2, and R3, respectively the first, second, and third rotation angles. The output is a 3-by-3 DCM that performs coordinate transformations based on rotation angles.



## Dialog Box



### Rotation Order

Specifies the input rotation order for three rotation angles. From the list, select ZYX, ZYZ, ZXY, ZXZ, YXZ, YXY, YZX, YZY, XYZ, XYX, XZY, or XZX. The default is ZYX.

## Inputs and Outputs

Input	Dimension Type	Description
First	3-by-1 vector	Contains the rotation angles, in radians.



# Rotation Angles to Direction Cosine Matrix

---

Output	Dimension Type	Description
First	3-by-3 matrix	Contains the direction cosine matrix.

## See Also

[Direction Cosine Matrix to Quaternions](#)

[Direction Cosine Matrix to Rotation Angles](#)

[Quaternions to Direction Cosine Matrix](#)

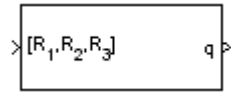
# Rotation Angles to Quaternions

---

**Purpose** Calculate quaternion from rotation angles

**Library** Utilities/Axes Transformations

**Description**



The Rotation Angles to Quaternions block converts the rotation described by the three rotation angles (R1, R2, R3) into the four-element quaternion vector  $(q_0, q_1, q_2, q_3)$ . A quaternion vector represents a

rotation about a unit vector  $(\mu_x, \mu_y, \mu_z)$  through an angle  $\theta$ . A unit quaternion itself has unit magnitude, and can be written in the following vector format.

$$q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos(\theta / 2) \\ \sin(\theta / 2)\mu_x \\ \sin(\theta / 2)\mu_y \\ \sin(\theta / 2)\mu_z \end{bmatrix}$$

An alternative representation of a quaternion is as a complex number,

$$q = q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3$$

where, for the purposes of multiplication,

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1$$

$$\mathbf{ij} = -\mathbf{ji} = \mathbf{k}$$

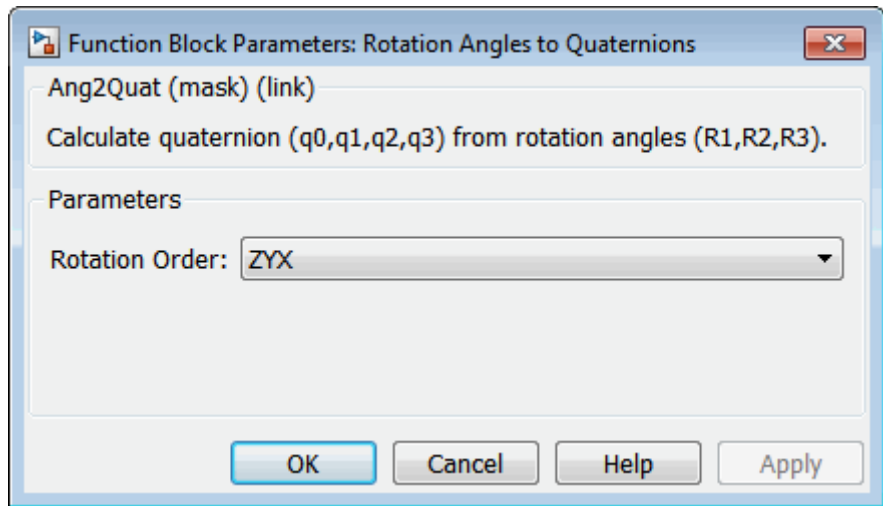
$$\mathbf{jk} = -\mathbf{kj} = \mathbf{i}$$

$$\mathbf{ki} = -\mathbf{ik} = \mathbf{j}$$

The benefit of representing the quaternion in this way is the ease with which the quaternion product can represent the resulting transformation after two or more rotations.

# Rotation Angles to Quaternions

## Dialog Box



### Rotation Order

Specifies the output rotation order for three wind rotation angles. From the list, select ZYX, ZYZ, ZXY, ZXZ, YXZ, YXY, YZX, YZY, XYZ, YXX, XZY, or XZX. The default is ZYX.

## Inputs and Outputs

Input	Dimension Type	Description
First	3-by-1 vector	Contains the rotation angles, in radians.

Output	Dimension Type	Description
First	4-by-1 matrix	Contains the quaternion vector.

# Rotation Angles to Quaternions

---

## **Assumptions and Limitations**

The limitations for the 'ZYX', 'ZXY', 'YXZ', 'YZX', 'XYZ', and 'XZY' implementations generate an R2 angle that is between  $\pm 90$  degrees, and R1 and R3 angles that are between  $\pm 180$  degrees.

The limitations for the 'ZYZ', 'ZXZ', 'YXY', 'YZY', 'YXX', and 'XZX' implementations generate an R2 angle that is between 0 and 180 degrees, and R1 and R3 angles that are between  $\pm 180$  degrees.

## **See Also**

[Direction Cosine Matrix to Quaternions](#)

[Quaternions to Direction Cosine Matrix](#)

[Quaternions to Rotation Angles](#)

[Rotation Angles to Direction Cosine Matrix](#)

## Purpose

Implement state-space controller in self-conditioned form

## Library

GNC/Controls

## Description



The Self-Conditioned [A,B,C,D] block can be used to implement the state-space controller defined by

$$\begin{bmatrix} \dot{x} = Ax + Be \\ u = Cx + De \end{bmatrix}$$

in the self-conditioned form

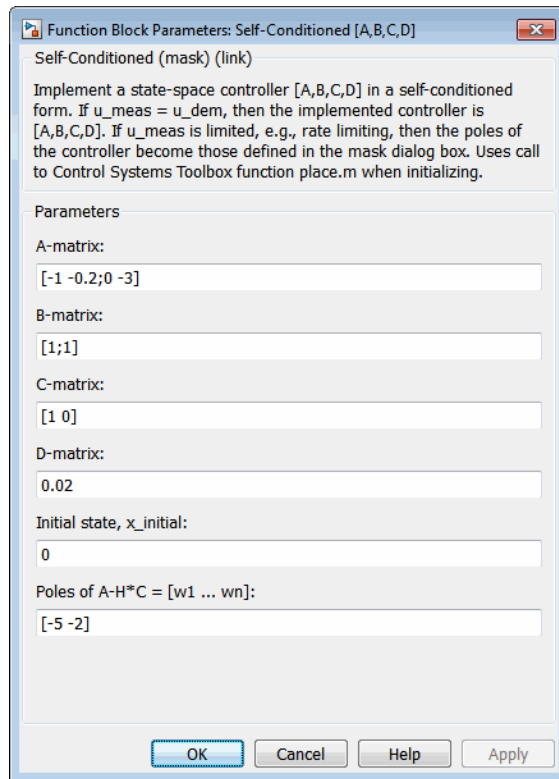
$$\begin{aligned} \dot{z} &= (A - HC)z + (B - HD)e + Hu_{meas} \\ u_{dem} &= Cz + De \end{aligned}$$

The input  $u_{meas}$  is a vector of the achieved actuator positions, and the output  $u_{dem}$  is the vector of controller actuator demands. In the case that the actuators are not limited, then  $u_{meas} = u_{dem}$  and substituting the output equation into the state equation returns the nominal controller. In the case that they are not equal, the dynamics of the controller are set by the poles of  $A-HC$ .

Hence  $H$  must be chosen to make the poles sufficiently fast to track  $u_{meas}$  but at the same time not so fast that noise on  $e$  is propagated to  $u_{dem}$ . The matrix  $H$  is designed by a callback to the Control System Toolbox command `place` to place the poles at defined locations.

# Self-Conditioned [A,B,C,D]

## Dialog Box



### A-matrix

A-matrix of the state-space implementation.

### B-matrix

B-matrix of the state-space implementation.

### C-matrix

C-matrix of the state-space implementation.

### D-matrix

D-matrix of the state-space implementation.

## Initial state, $x_{initial}$

This is a vector of initial states for the controller, i.e., initial values for the state vector,  $z$ . It should have length equal to the size of the first dimension of  $A$ .

## Poles of $A-H*C$

This is a vector of the desired poles of  $A-H*C$ . Hence the number of pole locations defined should be equal to the dimension of the  $A$ -matrix.

## Inputs and Outputs

Input	Dimension	Type	Description
First			Contains the control error.
Second			Contains the measured actuator position.

Output	Dimension	Type	Description
First			Contains the actuator demands.

## Assumptions and Limitations

---

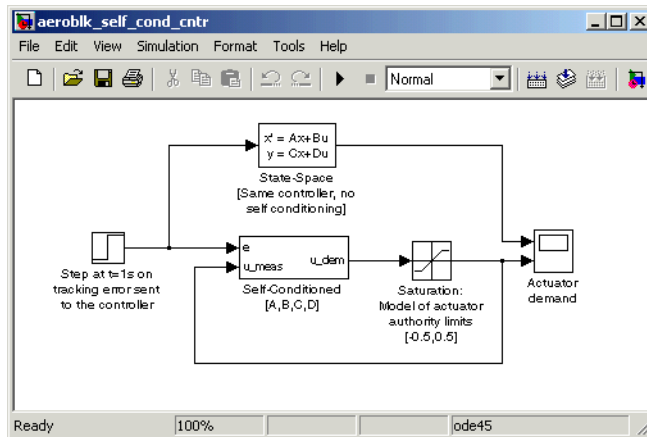
**Note** This block requires the Control System Toolbox product.

---

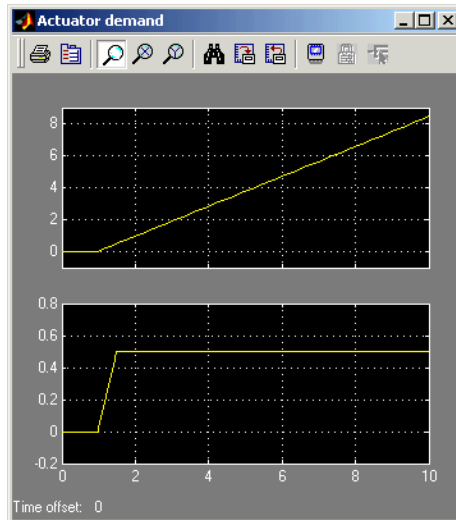
## Examples

This Simulink model shows a state-space controller implemented in both self-conditioned and standard state-space forms. The actuator authority limits of  $\pm 0.5$  units are modeled by the saturation block.

# Self-Conditioned [A,B,C,D]



Notice that the  $A$ -matrix has a zero in the 1,1 element, indicating integral action.



The top trace shows the conventional state-space implementation. The output of the controller winds up well past the actuator upper authority limit of +0.5. The lower trace shows that the self-conditioned form



results in an actuator demand that tracks the upper authority limit, which means that when the sign of the control error,  $e$ , is reversed, the actuator demand responds immediately.

## Reference

The algorithm used to determine the matrix  $H$  is defined in Kautsky, Nichols, and Van Dooren, "Robust Pole Assignment in Linear State Feedback," *International Journal of Control*, Vol. 41, No. 5, pages 1129-1155, 1985.

## See Also

1D Self-Conditioned [A(v),B(v),C(v),D(v)]

2D Self-Conditioned [A(v),B(v),C(v),D(v)]

3D Self-Conditioned [A(v),B(v),C(v),D(v)]

# Send net\_fdm Packet to FlightGear

---

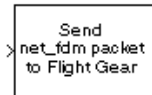
## Purpose

Transmit net\_fdm packet to destination IP address and port for FlightGear session

## Library

Animation/Flight Simulator Interfaces

## Description



The Send net\_fdm Packet to FlightGear block transmits the net\_fdm packet to FlightGear on the current computer, or a remote computer on the network. The packet is constructed using the Pack net\_fdm Packet for FlightGear block. The destination port should be an unused port that you can use when you launch FlightGear with the FlightGear command line flag:

```
--fdm=network,localhost,5501,5502,5503
```

The second port in the list, 5502, is the network flight dynamics model (fdm) port.

This block does not produce deployable code.

## Determining the Destination IP Address

You can use one of several techniques to determine the destination IP address, such as:

- Use 127.0.0.1 for “this” computer
- Ping another computer from a Windows cmd.exe (or UNIX shell) prompt:

```
C:\> ping andyspc
```

```
Pinging andyspc [144.213.175.92] with 32 bytes of data:
```

```
Reply from 144.213.175.92: bytes=32 time=30ms TTL=253
Reply from 144.213.175.92: bytes=32 time=20ms TTL=253
Reply from 144.213.175.92: bytes=32 time=20ms TTL=253
Reply from 144.213.175.92: bytes=32 time=20ms TTL=253
```

```
Ping statistics for 144.213.175.92:
```

# Send net\_fdm Packet to FlightGear

Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),  
Approximate round trip times in milli-seconds:  
Minimum = 20ms, Maximum = 30ms, Average = 22ms

- On a Windows machine, type `ipconfig` and use the returned *IP Address*:

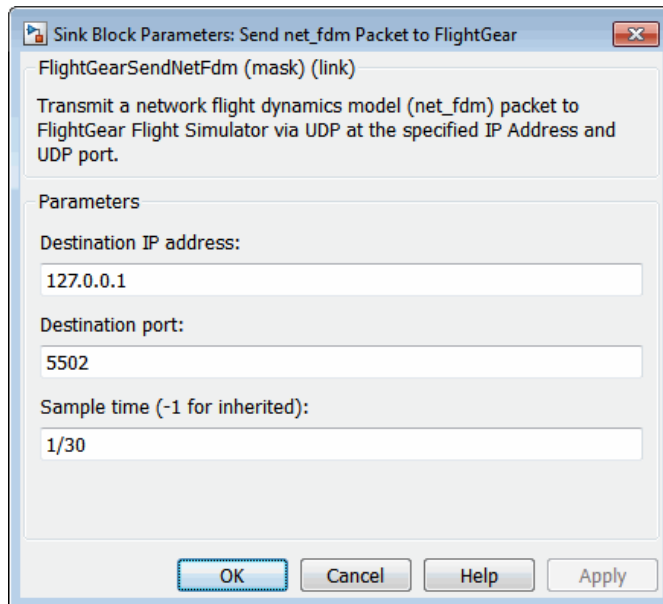
```
H:\>ipconfig
```

```
Windows IP Configuration
```

```
Ethernet adapter Local Area Connection:
```

```
Connection-specific DNS Suffix . . :  
IP Address. . . . . : 192.168.42.178  
Subnet Mask . . . . . : 255.255.255.0  
Default Gateway . . . . . : 192.168.42.254
```

## Dialog Box



# Send net\_fdm Packet to FlightGear

---

## **Destination IP address**

Specify your destination IP address.

## **Destination port**

Specify your destination port.

## **Sample time**

Specify the sample time (-1 for inherited).

## **Inputs and Outputs**

The input signal is the FlightGear net\_fdm data packet.

## **Examples**

See the asbh120 for an example of this block.

## **See Also**

FlightGear Preconfigured 6DoF Animation

Generate Run Script

Pack net\_fdm Packet for FlightGear

Receive net\_ctrl Packet from FlightGear

# Simple Variable Mass 3DoF (Body Axes)

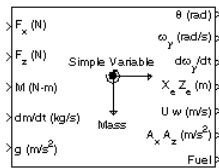
## Purpose

Implement three-degrees-of-freedom equations of motion of simple variable mass with respect to body axes

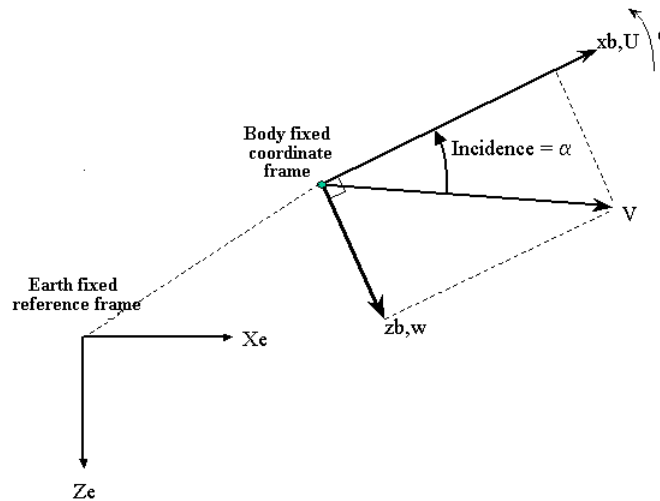
## Library

Equations of Motion/3DoF

## Description



The Simple Variable Mass 3DoF (Body Axes) block considers the rotation in the vertical plane of a body-fixed coordinate frame about an Earth-fixed reference frame.



The equations of motion are

## Simple Variable Mass 3DoF (Body Axes)

---

$$\dot{u} = \frac{F_x}{m} - \frac{\dot{m}U}{m} - qw - g \sin \theta$$

$$\dot{w} = \frac{F_z}{m} - \frac{\dot{m}w}{m} + qu + g \cos \theta$$

$$\dot{q} = \frac{M - \dot{I}_{yy}q}{I_{yy}}$$

$$\dot{\theta} = q$$

$$\dot{I}_{yy} = \frac{I_{yyfull} - I_{yyempty}}{m_{full} - m_{empty}} \dot{m}$$

where the applied forces are assumed to act at the center of gravity of the body.

# Simple Variable Mass 3DoF (Body Axes)

## Dialog Box

Function Block Parameters: Simple Variable Mass 3DoF (Body Axes)

3DoF EoM (mask) (link)

Integrate the three-degrees-of-freedom equations of motion to determine body position, velocity, attitude, and related values.

Parameters

Units: Metric (MKS)

Mass type: Simple Variable

Initial velocity: 100

Initial body attitude: 0

Initial incidence: 0

Initial body rotation rate: 0

Initial position (x z): [0 0]

Initial mass: 1.0

Empty mass: 0.5

Full mass: 3.0

Empty inertia: 0.5

Full inertia: 3.0

Gravity source: External

OK Cancel Help Apply

### Units

Specifies the input and output units:

# Simple Variable Mass 3DoF (Body Axes)

---

Units	Forces	Moment	Acceleration	Velocity	Position	Mass	Inertia
Metric (MKS)	Newton	Newton meter	Meters per second squared	Meters per second	Meters	Kilogram	Kilogram meter squared
English (Velocity in ft/s)	Pound	Foot pound	Feet per second squared	Feet per second	Feet	Slug	Slug foot squared
English (Velocity in kts)	Pound	Foot pound	Feet per second squared	Knots	Feet	Slug	Slug foot squared

## Mass Type

Select the type of mass to use:

Fixed	Mass is constant throughout the simulation.
Simple Variable	Mass and inertia vary linearly as a function of mass rate.
Custom Variable	Mass and inertia variations are customizable.

The Simple Variable selection conforms to the previously described equations of motion.

## Initial velocity

A scalar value for the initial velocity of the body, ( $V_0$ ).

## Initial body attitude

A scalar value for the initial pitch attitude of the body, ( $\theta_0$ ).

## Initial incidence

A scalar value for the initial angle between the velocity vector and the body, ( $\alpha_0$ ).



# Simple Variable Mass 3DoF (Body Axes)

---

## **Initial body rotation rate**

A scalar value for the initial body rotation rate, ( $q_0$ ).

## **Initial position (x,z)**

A two-element vector containing the initial location of the body in the Earth-fixed reference frame.

## **Initial mass**

A scalar value for the initial mass of the body.

## **Empty mass**

A scalar value for the empty mass of the body.

## **Full mass**

A scalar value for the full mass of the body.

## **Empty inertia**

A scalar value for the empty inertia of the body.

## **Full inertia**

A scalar value for the full inertia of the body.

## **Gravity source**

Specify source of gravity:

External

Variable gravity input to block

Internal

Constant gravity specified in mask

## **Acceleration due to gravity**

A scalar value for the acceleration due to gravity used if internal gravity source is selected. If gravity is to be neglected in the simulation, this value can be set to 0.

# Simple Variable Mass 3DoF (Body Axes)

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the force acting along the body $x$ -axis, ( $F_x$ ).
Second		Contains the force acting along the body $z$ -axis, ( $F_z$ ).
Third		Contains the applied pitch moment, ( $M$ ).
Fourth		Contains the rate of change of mass, ( $\dot{m}$ ).
Fifth (Optional)		Contains the gravity in the selected units.

Output	Dimension Type	Description
First		Contains the pitch attitude, in radians ( $\theta$ ).
Second		Contains the pitch angular rate, in radians per second ( $q$ ).
Third		Contains the pitch angular acceleration, in radians per second squared ( $\dot{q}$ ).
Fourth	Two-element vector	Contains the location of the body, in the Earth-fixed reference frame, ( $X_e$ , $Z_e$ ).
Fifth	Two-element vector	Contains the velocity of the body resolved into the body-fixed coordinate frame, ( $u$ , $w$ ).

# Simple Variable Mass 3DoF (Body Axes)

---

Output	Dimension Type	Description
Sixth	Two-element vector	Contains the acceleration of the body resolved into the body-fixed coordinate frame, $(A_x, A_z)$ .
Seventh	Scalar element	Contains a flag for fuel tank status, $(Fuel)$ : <ul style="list-style-type: none"><li>• 1 indicates that the tank is full.</li><li>• 0 indicates that the integral is neither full nor empty.</li><li>• -1 indicates that the tank is empty.</li></ul>

## See Also

[3DoF \(Body Axes\)](#)

[3DoF \(Wind Axes\)](#)

[Custom Variable Mass 3DoF \(Body Axes\)](#)

[Custom Variable Mass 3DoF \(Wind Axes\)](#)

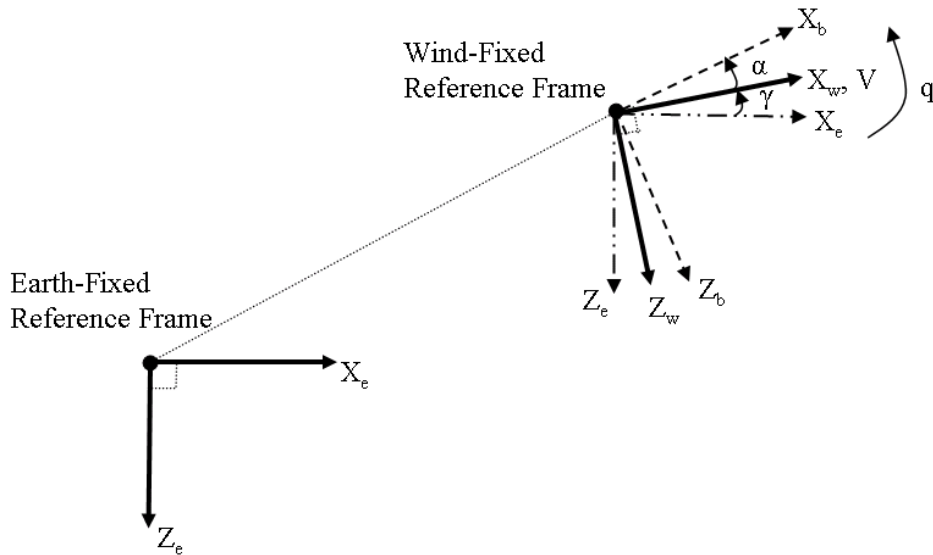
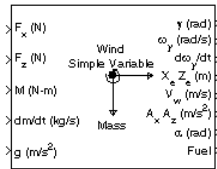
[Simple Variable Mass 3DoF \(Wind Axes\)](#)

# Simple Variable Mass 3DoF (Wind Axes)

**Purpose** Implement three-degrees-of-freedom equations of motion of simple variable mass with respect to wind axes

**Library** Equations of Motion/3DoF

**Description** The Simple Variable Mass 3DoF (Wind Axes) block considers the rotation in the vertical plane of a wind-fixed coordinate frame about an Earth-fixed reference frame.



The equations of motion are

## Simple Variable Mass 3DoF (Wind Axes)

---

$$\dot{V} = \frac{F_{x_{wind}}}{m} - \frac{\dot{m}V}{m} - g \sin \gamma$$

$$\dot{\alpha} = \frac{F_{z_{wind}}}{mV} + q + \frac{g}{V} \cos \gamma$$

$$\dot{q} = \dot{\theta} = \frac{M_{y_{body}} - \dot{I}_{yy}q}{I_{yy}}$$

$$\dot{\gamma} = q - \dot{\alpha}$$

$$\dot{I}_{yy} = \frac{I_{yyfull} - I_{yyempty}}{m_{full} - m_{empty}} \dot{m}$$

where the applied forces are assumed to act at the center of gravity of the body.

# Simple Variable Mass 3DoF (Wind Axes)

## Dialog Box

Function Block Parameters: Simple Variable Mass 3DoF (Wind Axes)

3DoF Wind EoM (mask) (link)

Integrate the three-degrees-of-freedom equations of motion in wind axes to determine position, velocity, attitude, and related values.

Parameters

Units: Metric (MKS)

Mass type: Simple Variable

Initial airspeed: 100

Initial flight path angle: 0

Initial incidence: 0

Initial body rotation rate: 0

Initial position (x z): [0 0]

Initial mass: 1.0

Empty mass: 0.5

Full mass: 3.0

Empty inertia body axes: 0.5

Full inertia body axes: 3.0

Gravity source: External

OK Cancel Help Apply

# Simple Variable Mass 3DoF (Wind Axes)

## Units

Specifies the input and output units:

<b>Units</b>	<b>Forces</b>	<b>Moment</b>	<b>Acceleration</b>	<b>Velocity</b>	<b>Position</b>	<b>Mass</b>	<b>Inertia</b>
Metric (MKS)	Newton	Newton meter	Meters per second squared	Meters per second	Meters	Kilogram	Kilogram meter squared
English (Velocity in ft/s)	Pound	Foot pound	Feet per second squared	Feet per second	Feet	Slug	Slug foot squared
English (Velocity in kts)	Pound	Foot pound	Feet per second squared	Knots	Feet	Slug	Slug foot squared

## Mass Type

Select the type of mass to use:

Fixed

Mass is constant throughout the simulation.

Simple Variable

Mass and inertia vary linearly as a function of mass rate.

Custom Variable

Mass and inertia variations are customizable.

The Simple Variable selection conforms to the previously described equations of motion.

## Initial airspeed

A scalar value for the initial velocity of the body, ( $V_0$ ).

## Initial flight path angle

A scalar value for the initial flight path angle of the body, ( $\gamma_0$ ).

# Simple Variable Mass 3DoF (Wind Axes)

---

## **Initial incidence**

A scalar value for the initial angle between the velocity vector and the body, ( $\alpha_0$ ).

## **Initial body rotation rate**

A scalar value for the initial body rotation rate, ( $q_0$ ).

## **Initial position (x,z)**

A two-element vector containing the initial location of the body in the Earth-fixed reference frame.

## **Initial mass**

A scalar value for the initial mass of the body.

## **Empty mass**

A scalar value for the empty mass of the body.

## **Full mass**

A scalar value for the full mass of the body.

## **Empty inertia**

A scalar value for the empty inertia of the body.

## **Full inertia**

A scalar value for the full inertia of the body.

## **Gravity source**

Specify source of gravity:

External

Variable gravity input to block

Internal

Constant gravity specified in mask

## **Acceleration due to gravity**

A scalar value for the acceleration due to gravity used if internal gravity source is selected. If gravity is to be neglected in the simulation, this value can be set to 0.



# Simple Variable Mass 3DoF (Wind Axes)

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the force acting along the wind $x$ -axis, ( $F_x$ ).
Second		Contains the force acting along the wind $z$ -axis, ( $F_z$ ).
Third		Contains the applied pitch moment in body axes, ( $M$ ).
Fourth		Contains the rate of change of mass, ( $\dot{m}$ ).
Fifth (Optional)		Contains the gravity in the selected units.

Output	Dimension Type	Description
First		Contains the flight path angle, in radians ( $\gamma$ ).
Second		Contains the pitch angular rate, in radians per second ( $\omega_y$ ).
Third		Contains the pitch angular acceleration, in radians per second squared ( $d\omega_y/dt$ ).
Fourth	Two-element vector	Contains the location of the body, in the Earth-fixed reference frame, ( $X_e, Z_e$ ).
Fifth	Two-element vector	Contains the velocity of the body resolved into the wind-fixed coordinate frame, ( $V, \theta$ ).
Sixth	Two-element vector	Contains the acceleration of the body resolved into the body-fixed coordinate frame, ( $A_x, A_z$ ).

# Simple Variable Mass 3DoF (Wind Axes)

---

Output Dimension Type	Description
Seventh Scalar	Contain the angle of attack, ( $\alpha_0$ ).
Eight Scalar element	Contains a flag for fuel tank status, ( <i>Fuel</i> ): <ul style="list-style-type: none"><li>• 1 indicates that the tank is full.</li><li>• 0 indicates that the integral is neither full nor empty.</li><li>• -1 indicates that the tank is empty.</li></ul>

## Reference

Stevens, B. L., and F. L. Lewis, *Aircraft Control and Simulation*, John Wiley & Sons, New York, 1992.

## See Also

3DoF (Body Axes)

3DoF (Wind Axes)

Custom Variable Mass 3DoF (Body Axes)

Custom Variable Mass 3DoF (Wind Axes)

Simple Variable Mass 3DoF (Body Axes)

# Simple Variable Mass 6DoF (Euler Angles)

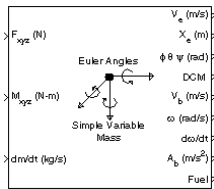
## Purpose

Implement Euler angle representation of six-degrees-of-freedom equations of motion of simple variable mass

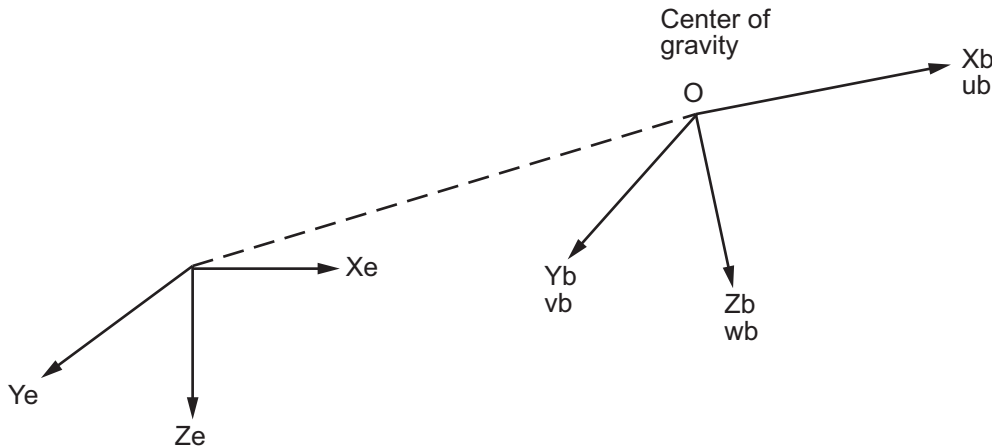
## Library

Equations of Motion/6DoF

## Description



The Simple Variable Mass 6DoF (Euler Angles) block considers the rotation of a body-fixed coordinate frame ( $X_b, Y_b, Z_b$ ) about a flat Earth reference frame ( $X_e, Y_e, Z_e$ ). The origin of the body-fixed coordinate frame is the center of gravity of the body, and the body is assumed to be rigid, an assumption that eliminates the need to consider the forces acting between individual elements of mass. The flat Earth reference frame is considered inertial, an excellent approximation that allows the forces due to the Earth's motion relative to the "fixed stars" to be neglected.



Flat Earth reference frame

The translational motion of the body-fixed coordinate frame is given below, where the applied forces  $[F_x \ F_y \ F_z]^T$  are in the body-fixed frame.

## Simple Variable Mass 6DoF (Euler Angles)

---

$$\bar{\mathbf{F}}_b = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = m(\dot{\bar{\mathbf{V}}}_b + \bar{\boldsymbol{\omega}} \times \bar{\mathbf{V}}_b) + \dot{m} \bar{\mathbf{V}}_b$$

$$\bar{\mathbf{V}}_b = \begin{bmatrix} u_b \\ v_b \\ w_b \end{bmatrix}, \bar{\boldsymbol{\omega}} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

The rotational dynamics of the body-fixed frame are given below, where the applied moments are  $[L \ M \ N]^T$ , and the inertia tensor  $I$  is with respect to the origin O.

$$\bar{\mathbf{M}}_B = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = I\dot{\bar{\boldsymbol{\omega}}} + \bar{\boldsymbol{\omega}} \times (I\bar{\boldsymbol{\omega}}) + \dot{I}\bar{\boldsymbol{\omega}}$$

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix}$$

The inertia tensor is determined using a table lookup which linearly interpolates between  $I_{full}$  and  $I_{empty}$  based on mass ( $m$ ). While the rate of change of the inertia tensor is estimated by the following equation.

$$\dot{I} = \frac{I_{full} - I_{empty}}{m_{full} - m_{empty}} \dot{m}$$

The relationship between the body-fixed angular velocity vector,  $[p \ q \ r]^T$ , and the rate of change of the Euler angles,  $[\dot{\phi} \ \dot{\theta} \ \dot{\psi}]^T$ , can be determined by resolving the Euler rates into the body-fixed coordinate frame.

# Simple Variable Mass 6DoF (Euler Angles)

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \equiv J^{-1} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

Inverting  $J$  then gives the required relationship to determine the Euler rate vector.

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = J \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & (\sin\phi \tan\theta) & (\cos\phi \tan\theta) \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

# Simple Variable Mass 6DoF (Euler Angles)

## Dialog Box

Function Block Parameters: Simple Variable Mass 6DoF (Euler Angles) ✕

6DoF EoM (Body Axis) (mask) (link)

Integrate the six-degrees-of-freedom equations of motion in body axis.

Parameters

Units:

Mass type:

Representation:

Initial position in inertial axes [Xe,Ye,Ze]:

Initial velocity in body axes [U,v,w]:

Initial Euler orientation [roll, pitch, yaw]:

Initial body rotation rates [p,q,r]:

Initial mass:

Empty mass:

Full mass:

Empty inertia matrix:

Full inertia matrix:

# Simple Variable Mass 6DoF (Euler Angles)

## Units

Specifies the input and output units:

<b>Units</b>	<b>Forces</b>	<b>Moment</b>	<b>Acceleration</b>	<b>Velocity</b>	<b>Position</b>	<b>Mass</b>	<b>Inertia</b>
Metric (MKS)	Newton	Newton meter	Meters per second squared	Meters per second	Meters	Kilogram	Kilogram meter squared
English (Velocity in ft/s)	Pound	Foot pound	Feet per second squared	Feet per second	Feet	Slug	Slug foot squared
English (Velocity in kts)	Pound	Foot pound	Feet per second squared	Knots	Feet	Slug	Slug foot squared

## Mass Type

Select the type of mass to use:

Fixed

Mass is constant throughout the simulation.

Simple Variable

Mass and inertia vary linearly as a function of mass rate.

Custom Variable

Mass and inertia variations are customizable.

The Simple Variable selection conforms to the previously described equations of motion.

## Representation

Select the representation to use:

# Simple Variable Mass 6DoF (Euler Angles)

---

Euler Angles

Use Euler angles within equations of motion.

Quaternion

Use quaternions within equations of motion.

The Euler Angles selection conforms to the previously described equations of motion.

## **Initial position in inertial axes**

The three-element vector for the initial location of the body in the flat Earth reference frame.

## **Initial velocity in body axes**

The three-element vector for the initial velocity in the body-fixed coordinate frame.

## **Initial Euler rotation**

The three-element vector for the initial Euler rotation angles [roll, pitch, yaw], in radians.

## **Initial body rotation rates**

The three-element vector for the initial body-fixed angular rates, in radians per second.

## **Initial mass**

The initial mass of the rigid body.

## **Empty mass**

A scalar value for the empty mass of the body.

## **Full mass**

A scalar value for the full mass of the body.

## **Empty inertia matrix**

A 3-by-3 inertia tensor matrix for the empty inertia of the body.

## **Full inertia matrix**

A 3-by-3 inertia tensor matrix for the full inertia of the body.



# Simple Variable Mass 6DoF (Euler Angles)

## Inputs and Outputs

Input	Dimension Type	Description
First	Vector	Contains the three applied forces.
Second	Vector	Contains the three applied moments.
Third	Scalar	Contains the rate of change of mass.

Output	Dimension Type	Description
First	Three-element vector	Contains the velocity in the flat Earth reference frame.
Second	Three-element vector	Contains the position in the flat Earth reference frame.
Third	Three-element vector	Contains the Euler rotation angles [roll, pitch, yaw], in radians.
Fourth	3-by-3 matrix	Applies to the coordinate transformation from flat Earth axes to body-fixed axes.
Fifth	Three-element vector	Contains the velocity in the body-fixed frame.
Sixth	Three-element vector	Contains the angular rates in body-fixed axes, in radians per second.
Seventh	Three-element vector	Contains the angular accelerations in body-fixed axes, in radians per second squared.

# Simple Variable Mass 6DoF (Euler Angles)

Output Dimension	Type	Description
Eight	Three-element vector	Contains the accelerations in body-fixed axes.
Ninth	Scalar element	Contains a flag for fuel tank status: <ul style="list-style-type: none"><li>• 1 indicates that the tank is full.</li><li>• 0 indicates that the integral is neither full nor empty.</li><li>• -1 indicates that the tank is empty.</li></ul>

## Assumptions and Limitations

The block assumes that the applied forces are acting at the center of gravity of the body.

## Reference

Mangiacasale, L., *Flight Mechanics of a  $\mu$ -Airplane with a MATLAB Simulink Helper*, Edizioni Libreria CLUP, Milan, 1998.

## See Also

6DoF (Euler Angles)  
6DoF (Quaternion)  
6DoF ECEF (Quaternion)  
6DoF Wind (Quaternion)  
6DoF Wind (Wind Angles)  
6th Order Point Mass (Coordinated Flight)  
Custom Variable Mass 6DoF (Euler Angles)  
Custom Variable Mass 6DoF (Quaternion)  
Custom Variable Mass 6DoF ECEF (Quaternion)  
Custom Variable Mass 6DoF Wind (Quaternion)  
Custom Variable Mass 6DoF Wind (Wind Angles)

# Simple Variable Mass 6DoF (Euler Angles)

---

Simple Variable Mass 6DoF (Quaternion)

Simple Variable Mass 6DoF ECEF (Quaternion)

Simple Variable Mass 6DoF Wind (Quaternion)

Simple Variable Mass 6DoF Wind (Wind Angles)

# Simple Variable Mass 6DoF (Quaternion)

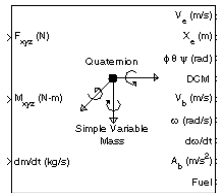
## Purpose

Implement quaternion representation of six-degrees-of-freedom equations of motion of simple variable mass with respect to body axes

## Library

Equations of Motion/6DoF

## Description



For a description of the coordinate system and the translational dynamics, see the block description for the Simple Variable Mass 6DoF (Euler Angles) block.

The integration of the rate of change of the quaternion vector is given below. The gain  $K$  drives the norm of the quaternion state vector to 1.0 should  $\varepsilon$  become nonzero. You must choose the value of this gain with care, because a large value improves the decay rate of the error in the norm, but also slows the simulation because fast dynamics are introduced. An error in the magnitude in one element of the quaternion vector is spread equally among all the elements, potentially increasing the error in the state vector.

$$\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} + K \varepsilon \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

$$\varepsilon = 1 - (q_0^2 + q_1^2 + q_2^2 + q_3^2)$$

# Simple Variable Mass 6DoF (Quaternion)

## Dialog Box

Function Block Parameters: Simple Variable Mass 6DoF (Quaternion) [X]

6DoF EoM (Body Axis) (mask) (link)

Integrate the six-degrees-of-freedom equations of motion in body axis.

Parameters

Units: Metric (MKS)

Mass type: Simple Variable

Representation: Quaternion

Initial position in inertial axes [Xe,Ye,Ze]:  
[0 0 0]

Initial velocity in body axes [U,v,w]:  
[0 0 0]

Initial Euler orientation [roll, pitch, yaw]:  
[0 0 0]

Initial body rotation rates [p,q,r]:  
[0 0 0]

Initial mass:  
1.0

Empty mass:  
0.5

Full mass:  
2.0

Empty inertia matrix:  
eye(3)

Full inertia matrix:  
2\*eye(3)

Gain for quaternion normalization:  
1.0

OK Cancel Help Apply

# Simple Variable Mass 6DoF (Quaternion)

---

## Units

Specifies the input and output units:

<b>Units</b>	<b>Forces</b>	<b>Moment</b>	<b>Acceleration</b>	<b>Velocity</b>	<b>Position</b>	<b>Mass</b>	<b>Inertia</b>
Metric (MKS)	Newton	Newton meter	Meters per second squared	Meters per second	Meters	Kilogram	Kilogram meter squared
English (Velocity in ft/s)	Pound	Foot pound	Feet per second squared	Feet per second	Feet	Slug	Slug foot squared
English (Velocity in kts)	Pound	Foot pound	Feet per second squared	Knots	Feet	Slug	Slug foot squared

## Mass Type

Select the type of mass to use:

Fixed	Mass is constant throughout the simulation.
Simple Variable	Mass and inertia vary linearly as a function of mass rate.
Custom Variable	Mass and inertia variations are customizable.

The Simple Variable selection conforms to the previously described equations of motion.

## Representation

Select the representation to use:

# Simple Variable Mass 6DoF (Quaternion)

---

Euler Angles	Use Euler angles within equations of motion.
Quaternion	Use quaternions within equations of motion.

The Quaternion selection conforms to the previously described equations of motion.

## **Initial position in inertial axes**

The three-element vector for the initial location of the body in the flat Earth reference frame.

## **Initial velocity in body axes**

The three-element vector for the initial velocity in the body-fixed coordinate frame.

## **Initial Euler rotation**

The three-element vector for the initial Euler rotation angles [roll, pitch, yaw], in radians.

## **Initial body rotation rates**

The three-element vector for the initial body-fixed angular rates, in radians per second.

## **Initial mass**

The initial mass of the rigid body.

## **Empty mass**

A scalar value for the empty mass of the body.

## **Full mass**

A scalar value for the full mass of the body.

## **Empty inertia matrix**

A 3-by-3 inertia tensor matrix for the empty inertia of the body.

## **Full inertia matrix**

A 3-by-3 inertia tensor matrix for the full inertia of the body.

# Simple Variable Mass 6DoF (Quaternion)

## Gain for quaternion normalization

The gain to maintain the norm of the quaternion vector equal to 1.0.

## Inputs and Outputs

Input	Dimension Type	Description
First	Vector	Contains the three applied forces.
Second	Vector	Contains the three applied moments.
Third	Scalar	Contains the rate of change of mass.

Output	Dimension Type	Description
First	Three-element vector	Contains the velocity in the flat Earth reference frame.
Second	Three-element vector	Contains the position in the flat Earth reference frame.
Third	Three-element vector	Contains the Euler rotation angles [roll, pitch, yaw], in radians.
Fourth	3-by-3 matrix	Applies to the coordinate transformation from flat Earth axes to body-fixed axes.
Fifth	Three-element vector	Contains the velocity in the body-fixed frame.
Sixth	Three-element vector	Contains the angular rates in body-fixed axes, in radians per second.
Seventh	Three-element vector	Contains the angular accelerations in body-fixed axes, in radians per second squared.



# Simple Variable Mass 6DoF (Quaternion)

Output Dimension	Type	Description
Eight	Three-element vector	Contains the accelerations in body-fixed axes.
Ninth	Scalar element	Contains a flag for fuel tank status: <ul style="list-style-type: none"><li>• 1 indicates that the tank is full.</li><li>• 0 indicates that the integral is neither full nor empty.</li><li>• -1 indicates that the tank is empty.</li></ul>

## Assumptions and Limitations

The block assumes that the applied forces are acting at the center of gravity of the body.

## Reference

Mangiacasale, L., *Flight Mechanics of a  $\mu$ -Airplane with a MATLAB Simulink Helper*, Edizioni Libreria CLUP, Milan, 1998.

## See Also

6DoF (Euler Angles)  
6DoF (Quaternion)  
6DoF ECEF (Quaternion)  
6DoF Wind (Quaternion)  
6DoF Wind (Wind Angles)  
6th Order Point Mass (Coordinated Flight)  
Custom Variable Mass 6DoF (Euler Angles)  
Custom Variable Mass 6DoF (Quaternion)  
Custom Variable Mass 6DoF ECEF (Quaternion)  
Custom Variable Mass 6DoF Wind (Quaternion)  
Custom Variable Mass 6DoF Wind (Wind Angles)

# Simple Variable Mass 6DoF (Quaternion)

---

Simple Variable Mass 6DoF (Euler Angles)

Simple Variable Mass 6DoF ECEF (Quaternion)

Simple Variable Mass 6DoF Wind (Quaternion)

Simple Variable Mass 6DoF Wind (Wind Angles)

# Simple Variable Mass 6DoF ECEF (Quaternion)

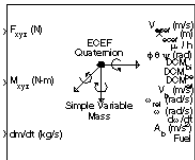
## Purpose

Implement quaternion representation of six-degrees-of-freedom equations of motion of simple variable mass in Earth-centered Earth-fixed (ECEF) coordinates

## Library

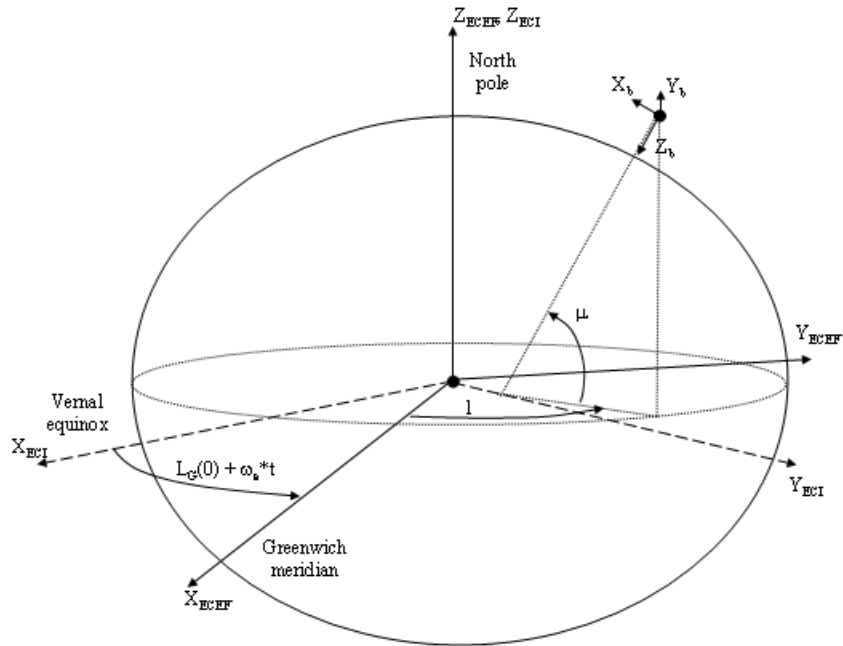
Equations of Motion/6DoF

## Description



The Simple Variable Mass 6DoF ECEF (Quaternion) block considers the rotation of a Earth-centered Earth-fixed (ECEF) coordinate frame ( $X_{ECEF}$ ,  $Y_{ECEF}$ ,  $Z_{ECEF}$ ) about an Earth-centered inertial (ECI) reference frame ( $X_{ECI}$ ,  $Y_{ECI}$ ,  $Z_{ECI}$ ). The origin of the ECEF coordinate frame is the center of the Earth, additionally the body of interest is assumed to be rigid, an assumption that eliminates the need to consider the forces acting between individual elements of mass. The representation of the rotation of ECEF frame from ECI frame is simplified to consider only the constant rotation of the ellipsoid Earth ( $\omega_e$ ) including an initial celestial longitude ( $L_G(0)$ ). This excellent approximation allows the forces due to the Earth's complex motion relative to the "fixed stars" to be neglected.

# Simple Variable Mass 6DoF ECEF (Quaternion)



The translational motion of the ECEF coordinate frame is given below, where the applied forces  $[F_x \ F_y \ F_z]^T$  are in the body frame.

$$\bar{F}_b = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = m \left( \dot{\bar{V}}_b + \bar{\omega}_b \times \bar{V}_b + DCM_{bf} \bar{\omega}_e \times \bar{V}_b \right) + DCM_{bf} \left( \bar{\omega}_e \times (\bar{\omega}_e \times \bar{X}_f) \right) + \dot{m} \left( \bar{V}_b + DCM_{bf} (\bar{\omega}_e \times \bar{X}_f) \right)$$

where the change of position in ECEF  $\dot{\bar{x}}_f(\dot{\bar{x}}_i)$  is calculated by

$$\dot{\bar{x}}_f = DCM_{fb} \bar{V}_b$$

and the velocity of the body with respect to ECEF frame, expressed in body frame  $(\bar{V}_b)$ , angular rates of the body with respect to ECI frame,

# Simple Variable Mass 6DoF ECEF (Quaternion)

expressed in body frame ( $\bar{\omega}_b$ ). Earth rotation rate ( $\bar{\omega}_e$ ), and relative angular rates of the body with respect to north-east-down (NED) frame, expressed in body frame ( $\bar{\omega}_{rel}$ ) are defined as

$$\bar{V}_b = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \quad \bar{\omega}_{rel} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad \bar{\omega}_e = \begin{bmatrix} 0 \\ 0 \\ \omega_e \end{bmatrix}$$

$$\bar{\omega}_b = \bar{\omega}_{rel} + DCM_{bf} \bar{\omega}_e + DCM_{be} \bar{\omega}_{ned}$$

$$\bar{\omega}_{ned} = \begin{bmatrix} \dot{\mu} \cos \mu \\ -\dot{\mu} \\ -\dot{\mu} \sin \mu \end{bmatrix} = \begin{bmatrix} V_E / (N + h) \\ -V_N / (M + h) \\ V_E \tan \mu / (N + h) \end{bmatrix}$$

The rotational dynamics of the body defined in body-fixed frame are given below, where the applied moments are  $[L \ M \ N]^T$ , and the inertia tensor  $I$  is with respect to the origin O.

$$\bar{M}_b = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = \bar{I} \dot{\bar{\omega}}_b + \bar{\omega}_b \times (\bar{I} \bar{\omega}_b) + \dot{\bar{I}} \bar{\omega}_b$$

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix}$$

The inertia tensor is determined using a table lookup which linearly interpolates between  $I_{full}$  and  $I_{empty}$  based on mass ( $m$ ). The rate of change of the inertia tensor is estimated by the following equation.

## Simple Variable Mass 6DoF ECEF (Quaternion)

---

$$\dot{I} = \frac{I_{full} - I_{empty}}{m_{full} - m_{empty}} \dot{m}$$

The integration of the rate of change of the quaternion vector is given below.

$$\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = -\frac{1}{2} \begin{bmatrix} 0 & \omega_b(1) & \omega_b(2) & \omega_b(3) \\ -\omega_b(1) & 0 & -\omega_b(3) & \omega_b(2) \\ -\omega_b(2) & \omega_b(3) & 0 & -\omega_b(1) \\ -\omega_b(3) & -\omega_b(2) & \omega_b(1) & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

# Simple Variable Mass 6DoF ECEF (Quaternion)

## Dialog Box

Function Block Parameters: Simple Variable Mass 6DoF ECEF (Quat...)

6DoF EoM (ECEF) (mask) (link)

Integrate the six-degrees-of-freedom equations of motion using a quaternion representation for the orientation of the body in space.

Main Planet

Units: Metric (MKS)

Mass type: Simple Variable

Initial position in geodetic latitude, longitude, altitude [mu,l,h]:  
[0 0 0]

Initial velocity in body axes [U,v,w]:  
[0 0 0]

Initial Euler orientation [roll, pitch, yaw]:  
[0 0 0]

Initial body rotation rates [p,q,r]:  
[0 0 0]

Initial mass:  
1.0

Empty mass:  
0.5

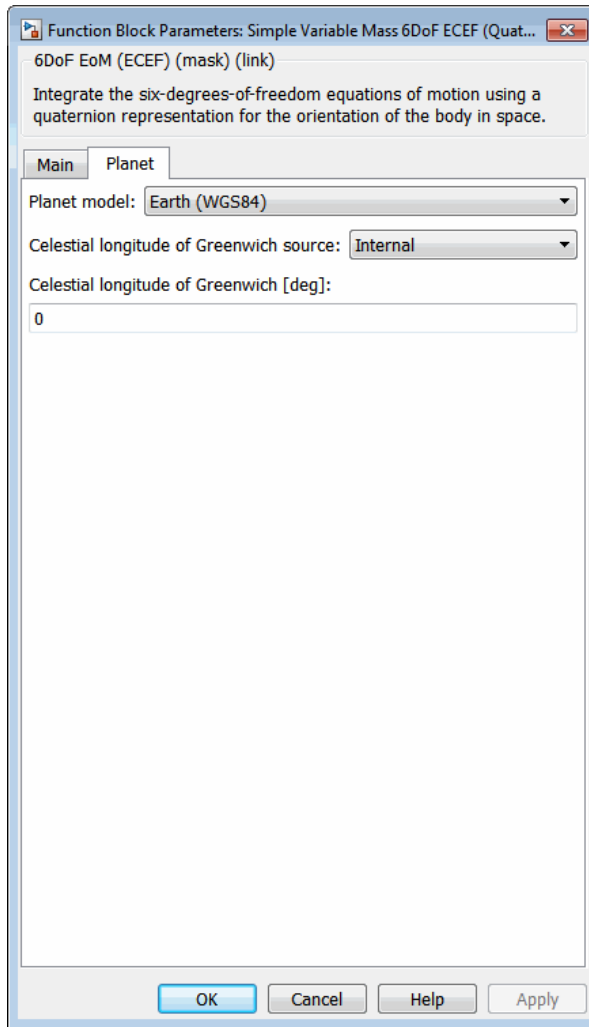
Full mass:  
2.0

Empty inertia matrix:  
eye(3)

Full inertia matrix:  
2\*eye(3)

OK Cancel Help Apply

# Simple Variable Mass 6DoF ECEF (Quaternion)



## Units

Specifies the input and output units:



# Simple Variable Mass 6DoF ECEF (Quaternion)

Units	Forces	Moment	Acceleration	Velocity	Position	Mass	Inertia
Metric (MKS)	Newton	Newton meter	Meters per second squared	Meters per second	Meters	Kilogram	Kilogram meter squared
English (Velocity in ft/s)	Pound	Foot pound	Feet per second squared	Feet per second	Feet	Slug	Slug foot squared
English (Velocity in kts)	Pound	Foot pound	Feet per second squared	Knots	Feet	Slug	Slug foot squared

## Mass type

Select the type of mass to use:

Fixed

Mass is constant throughout the simulation (see 6DoF ECEF (Quaternion)).

Simple Variable

Mass and inertia vary linearly as a function of mass rate.

Custom Variable

Mass and inertia variations are customizable (see Custom Variable Mass 6DoF ECEF (Quaternion)).

The Simple Variable selection conforms to the previously described equations of motion.

## Initial position in geodetic latitude, longitude and altitude

The three-element vector for the initial location of the body in the geodetic reference frame.

## Initial velocity in body axes

The three-element vector containing the initial velocity of the body with respect to the ECEF frame, expressed in the body frame.

# Simple Variable Mass 6DoF ECEF (Quaternion)

---

## **Initial Euler orientation**

The three-element vector containing the initial Euler rotation angles [roll, pitch, yaw], in radians. Euler rotation angles are those between the body and NED coordinate systems.

## **Initial body rotation rates**

The three-element vector for the initial angular rates of the body with respect to the NED frame, expressed the body frame, in radians per second.

## **Initial mass**

The mass of the rigid body.

## **Empty mass**

A scalar value for the empty mass of the body.

## **Full mass**

A scalar value for the full mass of the body.

## **Empty inertia matrix**

A 3-by-3 inertia tensor matrix for the empty inertia of the body.

## **Full inertia matrix**

A 3-by-3 inertia tensor matrix for the full inertia of the body.

## **Planet model**

Specifies the planet model to use: Custom or Earth (WGS84).

## **Flattening**

Specifies the flattening of the planet. This option is only available when **Planet model** is set to Custom.

## **Equatorial radius of planet**

Specifies the radius of the planet at its equator. The units of the equatorial radius parameter should be the same as the units for ECEF position. This option is only available when **Planet model** is set to Custom.

## **Rotational rate**

Specifies the scalar rotational rate of the planet in rad/s. This option is only available when **Planet model** is set to Custom.

# Simple Variable Mass 6DoF ECEF (Quaternion)

## Celestial longitude of Greenwich source

Specifies the source of Greenwich meridian's initial celestial longitude:

Internal	Use celestial longitude value from mask dialog.
External	Use external input for celestial longitude value.

## Celestial longitude of Greenwich

The initial angle between Greenwich meridian and the  $x$ -axis of the ECI frame.

## Inputs and Outputs

Input	Dimension Type	Description
First	Vector	Contains the three applied forces in body-fixed axes.
Second	Vector	Contains the three applied moments in body-fixed axes.
Third	Scalar	Contains the rate of change of mass.

Output	Dimension Type	Description
First	Three-element vector	Contains the velocity of body respect to ECEF frame, expressed in ECEF frame.
Second	Three-element vector	Contains the position in the ECEF reference frame.
Third	Three-element vector	Contains the position in geodetic latitude, longitude and altitude, in degrees, degrees and selected units of length respectively.

## Simple Variable Mass 6DoF ECEF (Quaternion)

---

Output	Dimension Type	Description
Fourth	Three-element vector	Contains the body rotation angles [roll, pitch, yaw], in radians. Euler rotation angles are those between body and NED coordinate systems.
Fifth	3-by-3 matrix	Applies to the coordinate transformation from ECI axes to body-fixed axes.
Sixth	3-by-3 matrix	Applies to the coordinate transformation from NED axes to body-fixed axes.
Seventh	3-by-3 matrix	Applies to the coordinate transformation from ECEF axes to NED axes.
Eighth	Three-element vector	Contains the velocity of body with respect to ECEF frame, expressed in body frame.
Ninth	Three-element vector	Contains the relative angular rates of body with respect to NED frame, expressed in body frame, in radians per second.
Tenth	Three-element vector	Contains the angular rates of the body with respect to ECI frame, expressed in body frame, in radians per second.
Eleventh	Three-element vector	Contains the angular accelerations of the body with respect to ECI frame, expressed in body frame, in radians per second squared.

# Simple Variable Mass 6DoF ECEF (Quaternion)

Output	Dimension Type	Description
Twelfth	Three-element vector	Contains the accelerations in body-fixed axes.
Thirteenth	Scalar	Is an element containing a flag for fuel tank status: <ul style="list-style-type: none"><li>• 1 indicates that the tank is full.</li><li>• 0 indicates that the integral is neither full nor empty.</li><li>• -1 indicates that the tank is empty.</li></ul>

## Assumptions and Limitations

This implementation assumes that the applied forces are acting at the center of gravity of the body.

This implementation generates a geodetic latitude that lies between  $\pm 90$  degrees, and longitude that lies between  $\pm 180$  degrees. Additionally, the MSL altitude is approximate.

The Earth is assumed to be ellipsoidal. By setting flattening to 0.0, a spherical planet can be achieved. The Earth's precession, nutation, and polar motion are neglected. The celestial longitude of Greenwich is Greenwich Mean Sidereal Time (GMST) and provides a rough approximation to the sidereal time.

The implementation of the ECEF coordinate system assumes that the origin is at the center of the planet, the  $x$ -axis intersects the Greenwich meridian and the equator, the  $z$ -axis is the mean spin axis of the planet, positive to the north, and the  $y$ -axis completes the right-hand system.

The implementation of the ECI coordinate system assumes that the origin is at the center of the planet, the  $x$ -axis is the continuation of the line from the center of the Earth through the center of the Sun toward the vernal equinox, the  $z$ -axis points in the direction of the mean equatorial plane's north pole, positive to the north, and the  $y$ -axis completes the right-hand system.

# Simple Variable Mass 6DoF ECEF (Quaternion)

---

## References

Stevens, B. L., and F. L. Lewis, *Aircraft Control and Simulation, Second Edition*, John Wiley & Sons, New York, 2003.

McFarland, Richard E., *A Standard Kinematic Model for Flight simulation at NASA-Ames*, NASA CR-2497.

“Supplement to Department of Defense World Geodetic System 1984 Technical Report: Part I - Methods, Techniques and Data Used in WGS84 Development,” DMA TR8350.2-A.

## See Also

6DoF (Euler Angles)

6DoF (Quaternion)

6DoF ECEF (Quaternion)

6DoF Wind (Quaternion)

6DoF Wind (Wind Angles)

6th Order Point Mass (Coordinated Flight)

Custom Variable Mass 6DoF (Euler Angles)

Custom Variable Mass 6DoF (Quaternion)

Custom Variable Mass 6DoF ECEF (Quaternion)

Custom Variable Mass 6DoF Wind (Quaternion)

Custom Variable Mass 6DoF Wind (Wind Angles)

Simple Variable Mass 6DoF (Euler Angles)

Simple Variable Mass 6DoF (Quaternion)

Simple Variable Mass 6DoF Wind (Quaternion)

Simple Variable Mass 6DoF Wind (Wind Angles)

# Simple Variable Mass 6DoF Wind (Quaternion)

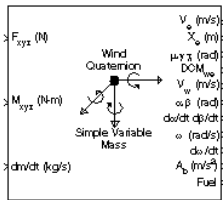
## Purpose

Implement quaternion representation of six-degrees-of-freedom equations of motion of simple variable mass with respect to wind axes

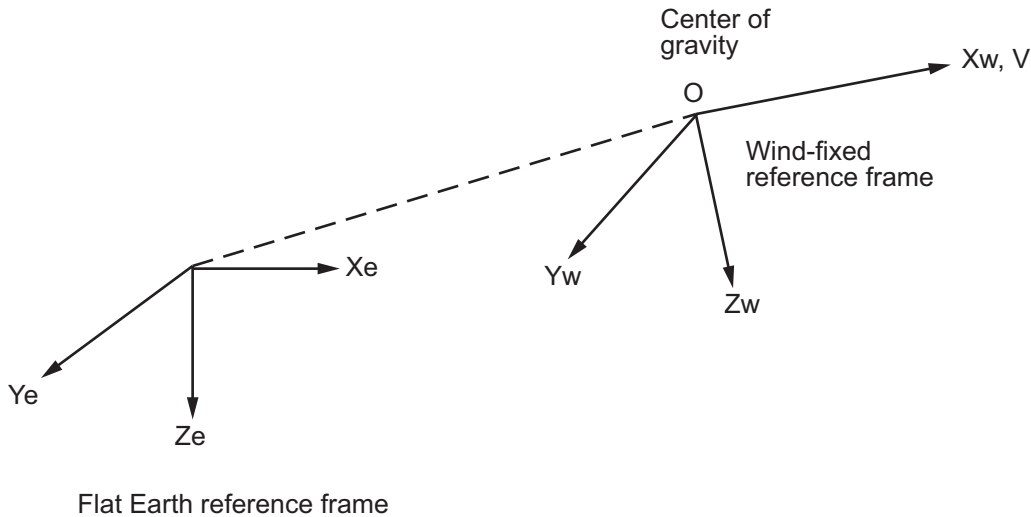
## Library

Equations of Motion/6DoF

## Description



The Simple Variable Mass 6DoF Wind (Quaternion) block considers the rotation of a wind-fixed coordinate frame ( $X_w, Y_w, Z_w$ ) about an flat Earth reference frame ( $X_e, Y_e, Z_e$ ). The origin of the wind-fixed coordinate frame is the center of gravity of the body, and the body is assumed to be rigid, an assumption that eliminates the need to consider the forces acting between individual elements of mass. The flat Earth reference frame is considered inertial, an excellent approximation that allows the forces due to the Earth's motion relative to the "fixed stars" to be neglected.



The translational motion of the wind-fixed coordinate frame is given below, where the applied forces  $[F_x \ F_y \ F_z]^T$  are in the wind-fixed frame.

## Simple Variable Mass 6DoF Wind (Quaternion)

$$\bar{\mathbf{F}}_w = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = m(\dot{\bar{\mathbf{V}}}_w + \bar{\omega}_w \times \bar{\mathbf{V}}_w) + \dot{m}\bar{\mathbf{V}}_w$$

$$\bar{\mathbf{V}}_w = \begin{bmatrix} V \\ 0 \\ 0 \end{bmatrix}, \bar{\omega}_w = \begin{bmatrix} p_w \\ q_w \\ r_w \end{bmatrix} = DMC_{wb} \begin{bmatrix} p_b - \dot{\beta} \sin \alpha \\ q_b - \dot{\alpha} \\ r_b + \dot{\beta} \cos \alpha \end{bmatrix}, \bar{\omega}_b = \begin{bmatrix} p_b \\ q_b \\ r_b \end{bmatrix}$$

The rotational dynamics of the body-fixed frame are given below, where the applied moments are  $[L \ M \ N]^T$ , and the inertia tensor  $I$  is with respect to the origin O. Inertia tensor  $I$  is much easier to define in body-fixed frame.

$$\bar{\mathbf{M}}_b = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = I\dot{\bar{\omega}}_b + \bar{\omega}_b \times (I\bar{\omega}_b) + \dot{I}\bar{\omega}_b$$

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix}$$

The inertia tensor is determined using a table lookup which linearly interpolates between  $I_{full}$  and  $I_{empty}$  based on mass ( $m$ ). While the rate of change of the inertia tensor is estimated by the following equation.

$$\dot{I} = \frac{I_{full} - I_{empty}}{m_{full} - m_{empty}} \dot{m}$$

The integration of the rate of change of the quaternion vector is given below.



# Simple Variable Mass 6DoF Wind (Quaternion)

---

$$\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = -1/2 \begin{bmatrix} 0 & p & q & r \\ -p & 0 & -r & q \\ -q & r & 0 & -p \\ -r & -q & p & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

# Simple Variable Mass 6DoF Wind (Quaternion)

## Dialog Box

Function Block Parameters: Simple Variable Mass 6DoF Wind (Quat... ✕

6DoF EoM (Wind Axis) (mask) (link)  
Integrate the six-degrees-of-freedom equations of motion in wind axis.

Parameters

Units:

Mass type:

Representation:

Initial position in inertial axes [Xe,Ye,Ze]:

Initial airspeed, angle of attack, and sideslip angle [V,alpha,beta]:

Initial wind orientation [bank angle,flight path angle,heading angle]:

Initial body rotation rates [p,q,r]:

Initial mass:

Empty mass:

Full mass:

Empty inertia matrix in body axis:

Full inertia matrix in body axis:

# Simple Variable Mass 6DoF Wind (Quaternion)

## Units

Specifies the input and output units:

Units	Forces	Moment	Acceleration	Velocity	Position	Mass	Inertia
Metric (MKS)	Newton	Newton meter	Meters per second squared	Meters per second	Meters	Kilogram	Kilogram meter squared
English (Velocity in ft/s)	Pound	Foot pound	Feet per second squared	Feet per second	Feet	Slug	Slug foot squared
English (Velocity in kts)	Pound	Foot pound	Feet per second squared	Knots	Feet	Slug	Slug foot squared

## Mass Type

Select the type of mass to use:

Fixed

Mass is constant throughout the simulation.

Simple Variable

Mass and inertia vary linearly as a function of mass rate.

Custom Variable

Mass and inertia variations are customizable.

The Simple Variable selection conforms to the previously described equations of motion.

## Representation

Select the representation to use:

# Simple Variable Mass 6DoF Wind (Quaternion)

---

Wind Angles

Use wind angles within equations of motion.

Quaternion

Use quaternions within equations of motion.

The Quaternion selection conforms to the previously described equations of motion.

## **Initial position in inertial axes**

The three-element vector for the initial location of the body in the flat Earth reference frame.

## **Initial airspeed, sideslip angle, and angle of attack**

The three-element vector containing the initial airspeed, initial sideslip angle and initial angle of attack.

## **Initial wind orientation**

The three-element vector containing the initial wind angles [bank, flight path, and heading], in radians.

## **Initial body rotation rates**

The three-element vector for the initial body-fixed angular rates, in radians per second.

## **Initial mass**

The initial mass of the rigid body.

## **Empty mass**

A scalar value for the empty mass of the body.

## **Full mass**

A scalar value for the full mass of the body.

## **Empty inertia matrix**

A 3-by-3 inertia tensor matrix for the empty inertia of the body, in body-fixed axes.

## **Full inertia matrix**

A 3-by-3 inertia tensor matrix for the full inertia of the body, in body-fixed axes.

# Simple Variable Mass 6DoF Wind (Quaternion)

## Inputs and Outputs

Input	Dimension Type	Description
First	Vector	Contains the three applied forces in wind-fixed axes.
Second	Vector	Contains the three applied moments in body-fixed axes.
Third	Scalar	Contains the rate of change of mass.

Output	Dimension Type	Description
First	Three-element vector	Contains the velocity in the flat Earth reference frame.
Second	Three-element vector	Contains the position in the flat Earth reference frame.
Third	Three-element vector	Contains the wind rotation angles [bank, flight path, heading], in radians.
Fourth	3-by-3 matrix	Applies to the coordinate transformation from flat Earth axes to wind-fixed axes.
Fifth	Three-element vector	Contains the velocity in the wind-fixed frame.
Sixth	Two-element vector	Contains the angle of attack and sideslip angle, in radians.
Seventh	Two-element vector	Contains the rate of change of angle of attack and rate of change of sideslip angle, in radians per second.
Eighth	Three-element vector	Contains the angular rates in body-fixed axes, in radians per second.

# Simple Variable Mass 6DoF Wind (Quaternion)

Output	Dimension Type	Description
Ninth	Three-element vector	Contains the angular accelerations in body-fixed axes, in radians per second squared.
Tenth	Three-element vector	Contains the accelerations in body-fixed axes.
Eleventh	Scalar element	Contains a flag for fuel tank status: <ul style="list-style-type: none"><li>• 1 indicates that the tank is full.</li><li>• 0 indicates that the integral is neither full nor empty.</li><li>• -1 indicates that the tank is empty.</li></ul>

## Assumptions and Limitations

The block assumes that the applied forces are acting at the center of gravity of the body.

## References

Mangiacasale, L., *Flight Mechanics of a  $\mu$ -Airplane with a MATLAB Simulink Helper*, Edizioni Libreria CLUP, Milan, 1998.

Stevens, B. L., and F. L. Lewis, *Aircraft Control and Simulation*, John Wiley & Sons, New York, 1992.

## See Also

6DoF (Euler Angles)

6DoF (Quaternion)

6DoF ECEF (Quaternion)

6DoF Wind (Quaternion)

6DoF Wind (Wind Angles)

6th Order Point Mass (Coordinated Flight)

Custom Variable Mass 6DoF (Euler Angles)

# Simple Variable Mass 6DoF Wind (Quaternion)

---

Custom Variable Mass 6DoF (Quaternion)

Custom Variable Mass 6DoF ECEF (Quaternion)

Custom Variable Mass 6DoF Wind (Quaternion)

Custom Variable Mass 6DoF Wind (Wind Angles)

Simple Variable Mass 6DoF (Euler Angles)

Simple Variable Mass 6DoF (Quaternion)

Simple Variable Mass 6DoF ECEF (Quaternion)

Simple Variable Mass 6DoF Wind (Wind Angles)

# Simple Variable Mass 6DoF Wind (Wind Angles)

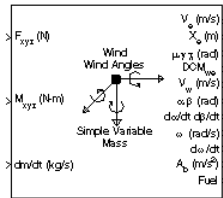
## Purpose

Implement wind angle representation of six-degrees-of-freedom equations of motion of simple variable mass

## Library

Equations of Motion/6DoF

## Description



For a description of the coordinate system employed and the translational dynamics, see the block description for the Simple Variable Mass 6DoF (Quaternion) block.

The relationship between the wind angles,  $[\mu \gamma \chi]^T$ , can be determined by resolving the wind rates into the wind-fixed coordinate frame.

$$\begin{bmatrix} p_w \\ q_w \\ r_w \end{bmatrix} = \begin{bmatrix} \dot{\mu} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \mu & \sin \mu \\ 0 & -\sin \mu & \cos \mu \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\gamma} \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \mu & \sin \mu \\ 0 & -\sin \mu & \cos \mu \end{bmatrix} \begin{bmatrix} \cos \gamma & 0 & -\sin \gamma \\ 0 & 1 & 0 \\ \sin \gamma & 0 & \cos \gamma \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\chi} \end{bmatrix} \equiv J^{-1} \begin{bmatrix} \dot{\mu} \\ \dot{\gamma} \\ \dot{\chi} \end{bmatrix}$$

Inverting  $J$  then gives the required relationship to determine the wind rate vector.

$$\begin{bmatrix} \dot{\mu} \\ \dot{\gamma} \\ \dot{\chi} \end{bmatrix} = J \begin{bmatrix} p_w \\ q_w \\ r_w \end{bmatrix} = \begin{bmatrix} 1 & (\sin \mu \tan \gamma) & (\cos \mu \tan \gamma) \\ 0 & \cos \mu & -\sin \mu \\ 0 & \frac{\sin \mu}{\cos \gamma} & \frac{\cos \mu}{\cos \gamma} \end{bmatrix} \begin{bmatrix} p_w \\ q_w \\ r_w \end{bmatrix}$$

The body-fixed angular rates are related to the wind-fixed angular rate by the following equation.

$$\begin{bmatrix} p_w \\ q_w \\ r_w \end{bmatrix} = DMC_{wb} \begin{bmatrix} p_b - \dot{\beta} \sin \alpha \\ q_b - \dot{\alpha} \\ r_b + \dot{\beta} \cos \alpha \end{bmatrix}$$



# Simple Variable Mass 6DoF Wind (Wind Angles)

---

Using this relationship in the wind rate vector equations, gives the relationship between the wind rate vector and the body-fixed angular rates.

$$\begin{bmatrix} \dot{\mu} \\ \dot{\gamma} \\ \dot{\chi} \end{bmatrix} = \mathbf{J} \begin{bmatrix} p_w \\ q_w \\ r_w \end{bmatrix} = \begin{bmatrix} 1 & (\sin \mu \tan \gamma) & (\cos \mu \tan \gamma) \\ 0 & \cos \mu & -\sin \mu \\ 0 & \frac{\sin \mu}{\cos \gamma} & \frac{\cos \mu}{\cos \gamma} \end{bmatrix} DMC_{wb} \begin{bmatrix} p_b - \dot{\beta} \sin \alpha \\ q_b - \dot{\alpha} \\ r_b + \dot{\beta} \cos \alpha \end{bmatrix}$$

# Simple Variable Mass 6DoF Wind (Wind Angles)

## Dialog Box

Function Block Parameters: Simple Variable Mass 6DoF Wind (Wind...)

6DoF EoM (Wind Axis) (mask) (link)

Integrate the six-degrees-of-freedom equations of motion in wind axis.

Parameters

Units: Metric (MKS)

Mass type: Simple Variable

Representation: Wind Angles

Initial position in inertial axes [Xe,Ye,Ze]:  
[0 0 0]

Initial airspeed, angle of attack, and sideslip angle [V,alpha,beta]:  
[0 0 0]

Initial wind orientation [bank angle,flight path angle,heading angle]:  
[0 0 0]

Initial body rotation rates [p,q,r]:  
[0 0 0]

Initial mass:  
1.0

Empty mass:  
0.5

Full mass:  
2.0

Empty inertia matrix in body axis:  
eye(3)

Full inertia matrix in body axis:  
2\*eye(3)

OK Cancel Help Apply

# Simple Variable Mass 6DoF Wind (Wind Angles)

## Units

Specifies the input and output units:

<b>Units</b>	<b>Forces</b>	<b>Moment</b>	<b>Acceleration</b>	<b>Velocity</b>	<b>Position</b>	<b>Mass</b>	<b>Inertia</b>
Metric (MKS)	Newton	Newton meter	Meters per second squared	Meters per second	Meters	Kilogram	Kilogram meter squared
English (Velocity in ft/s)	Pound	Foot pound	Feet per second squared	Feet per second	Feet	Slug	Slug foot squared
English (Velocity in kts)	Pound	Foot pound	Feet per second squared	Knots	Feet	Slug	Slug foot squared

## Mass Type

Select the type of mass to use:

Fixed

Mass is constant throughout the simulation.

Simple Variable

Mass and inertia vary linearly as a function of mass rate.

Custom Variable

Mass and inertia variations are customizable.

The Simple Variable selection conforms to the previously described equations of motion.

## Representation

Select the representation to use:

# Simple Variable Mass 6DoF Wind (Wind Angles)

---

Wind Angles

Use wind angles within equations of motion.

Quaternion

Use quaternions within equations of motion.

The Wind Angles selection conforms to the previously described equations of motion.

## **Initial position in inertial axes**

The three-element vector for the initial location of the body in the flat Earth reference frame.

## **Initial airspeed, sideslip angle, and angle of attack**

The three-element vector containing the initial airspeed, initial sideslip angle and initial angle of attack.

## **Initial wind orientation**

The three-element vector containing the initial wind angles [bank, flight path, and heading], in radians.

## **Initial body rotation rates**

The three-element vector for the initial body-fixed angular rates, in radians per second.

## **Initial mass**

The initial mass of the rigid body.

## **Empty mass**

A scalar value for the empty mass of the body.

## **Full mass**

A scalar value for the full mass of the body.

## **Empty inertia matrix**

A 3-by-3 inertia tensor matrix for the empty inertia of the body, in body-fixed axes.

## **Full inertia matrix**

A 3-by-3 inertia tensor matrix for the full inertia of the body, in body-fixed axes.

# Simple Variable Mass 6DoF Wind (Wind Angles)

## Inputs and Outputs

Input	Dimension Type	Description
First	Vector	Contains the three applied forces in wind-fixed axes.
Second	Vector	Contains the three applied moments in body-fixed axes.
Third	Scalar	Contains the rate of change of mass.

Output	Dimension Type	Description
First	Three-element vector	Contains the velocity in the fixed Earth reference frame.
Second	Three-element vector	Contains the position in the flat Earth reference frame.
Third	Three-element vector	Contains the wind rotation angles [bank, flight path, heading], in radians.
Fourth	3-by-3 matrix	Applies to the coordinate transformation from flat Earth axes to wind-fixed axes.
Fifth	Three-element vector	Contains the velocity in the wind-fixed frame.
Sixth	Two-element vector	Contains the angle of attack and sideslip angle, in radians.
Seventh	Two-element vector	Contains the rate of change of angle of attack and rate of change of sideslip angle, in radians per second.
Eighth	Three-element vector	Contains the angular rates in body-fixed axes, in radians per second.

# Simple Variable Mass 6DoF Wind (Wind Angles)

Output	Dimension Type	Description
Ninth	Three-element vector	Contain the angular accelerations in body-fixed axes, in radians per second squared.
Tenth	Three-element vector	Contains the accelerations in body-fixed axes.
Eleventh	Scalar element	Contains a flag for fuel tank status: <ul style="list-style-type: none"><li>• 1 indicates that the tank is full.</li><li>• 0 indicates that the integral is neither full nor empty.</li><li>• -1 indicates that the tank is empty.</li></ul>

## Assumptions and Limitations

The block assumes that the applied forces are acting at the center of gravity of the body.

## References

Mangiacasale, L., *Flight Mechanics of a  $\mu$ -Airplane with a MATLAB Simulink Helper*, Edizioni Libreria CLUP, Milan, 1998.

Stevens, B. L., and F. L. Lewis, *Aircraft Control and Simulation*, John Wiley & Sons, New York, 1992.

## See Also

6DoF (Euler Angles)

6DoF (Quaternion)

6DoF ECEF (Quaternion)

6DoF Wind (Quaternion)

6DoF Wind (Wind Angles)

6th Order Point Mass (Coordinated Flight)

Custom Variable Mass 6DoF (Euler Angles)

# Simple Variable Mass 6DoF Wind (Wind Angles)

---

Custom Variable Mass 6DoF (Quaternion)

Custom Variable Mass 6DoF ECEF (Quaternion)

Custom Variable Mass 6DoF Wind (Quaternion)

Custom Variable Mass 6DoF Wind (Wind Angles)

Simple Variable Mass 6DoF (Euler Angles)

Simple Variable Mass 6DoF (Quaternion)

Simple Variable Mass 6DoF ECEF (Quaternion)

Simple Variable Mass 6DoF Wind (Quaternion)

# Simulation Pace

---

## Purpose

Set simulation rate for improved animation viewing

## Library

Animation/Animation Support Utilities

## Description



The Simulation Pace block lets you run the simulation at the specified pace so that connected animations appear aesthetically pleasing.

This block does not produce deployable code.

Use the **Sample time** parameter to set how often the Simulink interface synchronizes with the wall clock.

The sample time of this block should be considered for human interaction with visualizations. The default is 1/30th of a second, chosen to correspond to a 30 frames-per-second visualization rate (typical for desktop computers).

---

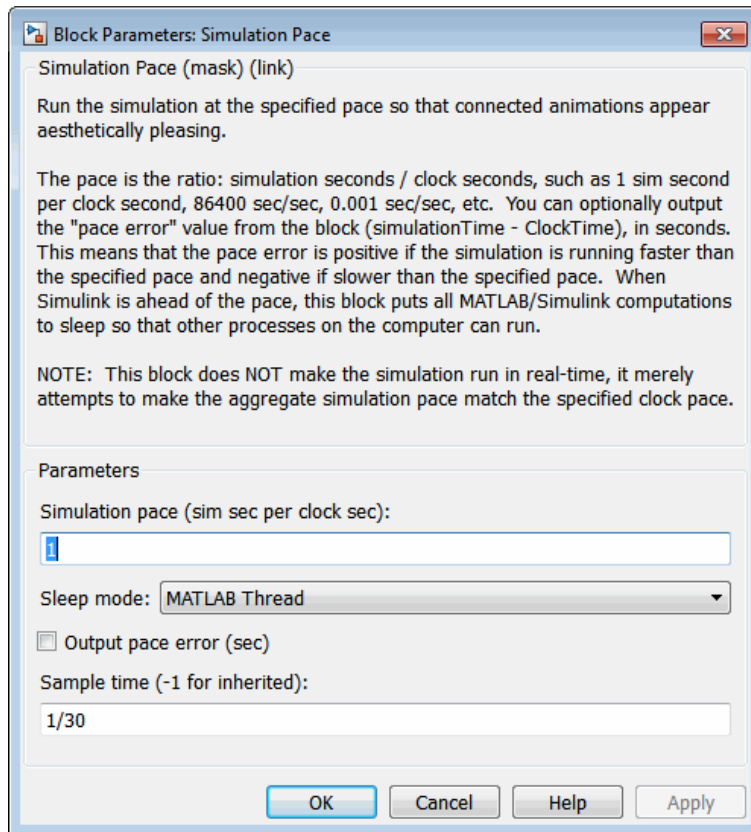
## Caution

Choose as slow of a sample time as needed for smooth animation, since oversampling has little benefit and undersampling can cause animation “jumpiness” and potentially block the MATLAB main thread on your computer.

---



## Dialog Box



### Simulation pace

Specifies the ratio of simulation time to clock time. The default is 1 second of simulation time per second of clock time.

### Sleep mode

Setting the **Sleep mode** parameter to off lets you disable the pace functionality and run as fast as possible.

### Output pace error

If you select this check box, the block outputs the "pace error" value (simulationTime minus ClockTime), in seconds. The pace

# Simulation Pace

---

error is positive if the simulation is running faster than the specified pace and negative if slower than the specified pace.

## Sample time

Specify the sample time (-1 for inherited). Larger sample times result in more efficient simulations, but less smooth in output pace when there are multiple Simulink time steps between pacer block samples. If the **Sample time** is too large, the MATLAB interface may become less responsive as MATLAB and Simulink calculations are blocked from running when the block puts the MATLAB interface to sleep.

## Inputs and Outputs

The block optionally outputs the “pace error” value (simulationTime minus ClockTime), in seconds. The pace error is positive if the simulation is running faster than the specified pace and negative if slower than the specified pace.

Outputting the pace error from the block lets you record the overall pace achieved during the simulation or routing the signal to other blocks to make decisions about the simulation if the simulation is too slow to keep up with the specified pace.

## Assumptions and Limitations

The simulation pace is implemented by putting the entire MATLAB thread to sleep until it needs to run again to keep up the pace. The Simulink software is single threaded and runs on the one MATLAB thread, so only one Simulation Pace block can be active at a time.

## Examples

See asbh120 for an example of this block.

## See Also

Pilot Joystick

**Purpose** Compute sine and cosine of angle

---

**Note** SinCos will be removed in a future release. Use the Simulink Trigonometric Function block instead.

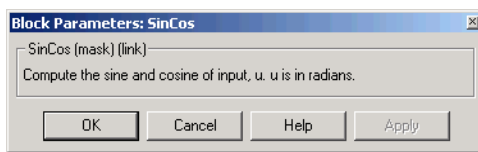
---

**Library** Utilities/Math Operations

**Description** The SinCos block computes the sine and cosine of the input angle, theta.



**Dialog Box**



**Inputs and Outputs**

The first input is an angle, in radians.

The first output is the sine of the input angle.

The second output is the cosine of the input angle.

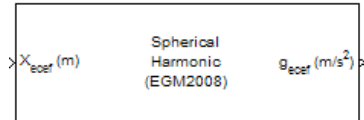
# Spherical Harmonic Gravity Model

---

**Purpose** Implement spherical harmonic representation of planetary gravity

**Library** Environment/Gravity

## Description



Spherical Harmonic Gravity Model

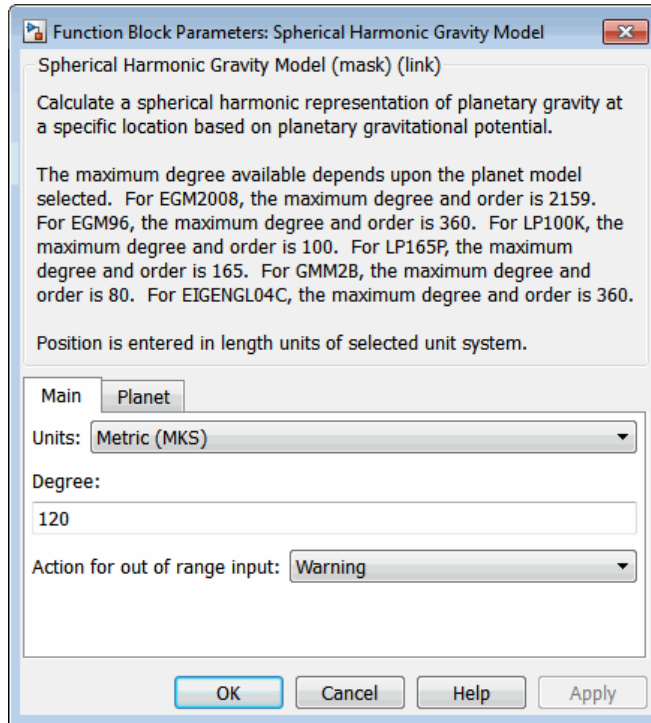
The Spherical Harmonic Gravity Model block implements the mathematical representation of spherical harmonic planetary gravity based on planetary gravitational potential. It provides a convenient way to describe a planet gravitational field outside of its surface in spherical harmonic expansion.

You can use spherical harmonics to modify the magnitude and direction of spherical gravity ( $-GM/r^2$ ). The most significant or largest spherical harmonic term is the second degree zonal harmonic,  $J_2$ , which accounts for oblateness of a planet.

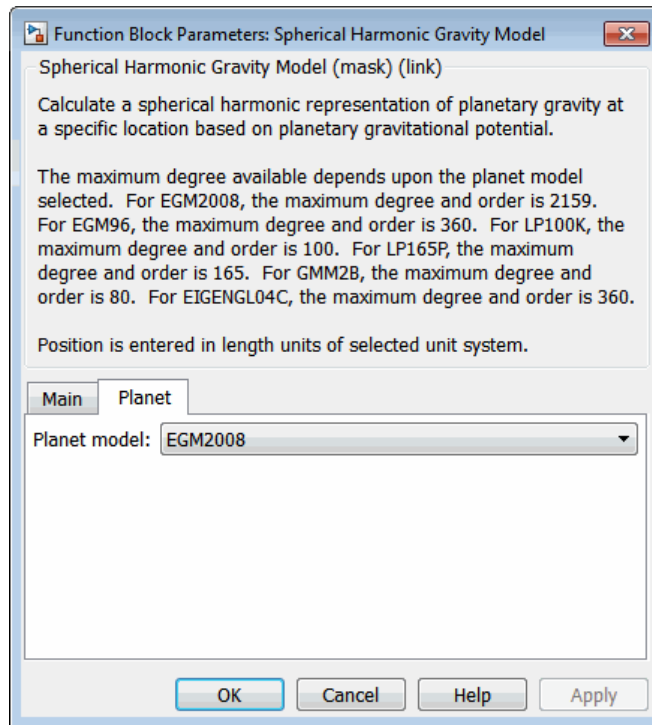
Use this block if you want more accurate gravity values than spherical gravity models. For example, nonatmospheric flight applications might require higher accuracy.

# Spherical Harmonic Gravity Model

## Dialog Box



# Spherical Harmonic Gravity Model



## Units

Specifies the parameter and output units:

<b>Units</b>	<b>Height</b>
Metric (MKS)	Meters
English	Feet

## Degree

Specify the degree of harmonic model. Recommended degrees are:

# Spherical Harmonic Gravity Model

<b>Planet Model</b>	<b>Degree</b>
EGM2008	120
EGM96	70
LP100K	60
LP165P	60
GMM2B	60
EIGENGL04C	70

## Action for out of range input

Specify if out-of-range input invokes a warning, error, or no action.

## Planet model

Specify the planetary model. From the list, select:

<b>Planet Model</b>	<b>Notes</b>
EGM2008	Earth — Is the latest Earth spherical harmonic gravitational model from National Geospatial-Intelligence Agency (NGA). This block provides the WGS-84 version of this gravitational model. You can use the EGM96 planetary model if you need to use the older standard for Earth.
EGM96	Earth
LP100K	Moon — Is best for lunar orbit determination based upon computational time required to compute orbits. This planet model was created in approximately the same year as LP165P with similar data.

# Spherical Harmonic Gravity Model

Planet Model	Notes
LP165P	Moon — Is best for extended lunar mission orbit accuracy. This planet model was created in approximately the same year as LP165P with similar data.
GMM2B	Mars
Custom	Enables you to specify your own planetary model. This option enables the <b>Planet mat-file</b> parameter.
EIGENGL04C	Earth — Supports the gravity field model, EIGEN-GL04C ( <a href="http://icgem.gfz-potsdam.de/ICGEM/">http://icgem.gfz-potsdam.de/ICGEM/</a> ). This model is an upgrade to EIGEN-CG03C.

When defining your own planetary model, the **Degree** parameter is limited to the maximum value for `int16`. When inputting a large degree, you might receive an out-of-memory error. For more information about avoiding out-of-memory errors in the MATLAB environment, see “Memory Usage”.

## Planet mat-file

Specify a MAT-file that contains definitions for a custom planetary model. The `aerogmm2b.mat` file in the Aerospace Toolbox is the default MAT-file for a custom planetary model.

This file must contain:

Variable	Description
<i>Re</i>	Scalar of planet equatorial radius in meters (m).
<i>GM</i>	Scalar of planetary gravitational parameter in meters cubed per second squared ( $\text{m}^3/\text{s}^2$ )
<i>degree</i>	Scalar of maximum degree.



# Spherical Harmonic Gravity Model

Variable	Description
$C$	$(degree+1)$ -by- $(degree+1)$ matrix containing normalized spherical harmonic coefficients matrix, $C$ .
$S$	$(degree+1)$ -by- $(degree+1)$ matrix containing normalized spherical harmonic coefficients matrix, $S$ .

When using a large value for **Degree**, you might receive an out-of-memory error. For more information about avoiding out-of-memory errors in the MATLAB environment, see “Memory Usage”.

## References

- [1] Gottlieb, R. G., “Fast Gravity, Gravity Partial, Normalized Gravity, Gravity Gradient Torque and Magnetic Field: Derivation, Code and Data,” *Technical Report NASA Contractor Report 188243*, NASA Lyndon B. Johnson Space Center, Houston, Texas, February 1993.
- [2] Vallado, D. A., *Fundamentals of Astrodynamics and Applications*, McGraw-Hill, New York, 1997.
- [3] “NIMA TR8350.2: Department of Defense World Geodetic System 1984, Its Definition and Relationship with Local Geodetic Systems”.
- [4] Konopliv, A. S., S. W. Asmar, E. Carranza, W. L. Sjogren, D. N. Yuan., “Recent Gravity Models as a Result of the Lunar Prospector Mission, Icarus”, Vol. 150, no. 1, pp 1–18, 2001.
- [5] Lemoine, F. G., D. E. Smith, D.D. Rowlands, M.T. Zuber, G. A. Neumann, and D. S. Chinn, “An improved solution of the gravity field of Mars (GMM-2B) from Mars Global Surveyor”, *Journal Of Geophysical Research*, Vol. 106, No. E10, pp 23359-23376, October 25, 2001.

# Spherical Harmonic Gravity Model

---

- [6] Kenyon S., J. Factor, N. Pavlis, and S. Holmes, “Towards the Next Earth Gravitational Model”, Society of Exploration Geophysicists 77th Annual Meeting, San Antonio, Texas, September 23–28, 2007.
- [7] Pavlis, N.K., S.A. Holmes, S.C. Kenyon, and J.K. Factor, “An Earth Gravitational Model to Degree 2160: EGM2008”, presented at the 2008 General Assembly of the European Geosciences Union, Vienna, Austria, April 13–18, 2008.
- [8] Grueber, T., and A. Köhl, “Validation of the EGM2008 Gravity Field with GPS-Leveling and Oceanographic Analyses”, presented at the IAG International Symposium on Gravity, Geoid & Earth Observation 2008, Chania, Greece, June 23–27, 2008.
- [9] Förste, C., Flechtner, F., Schmidt, R., König, R., Meyer, U., Stubenvoll, R., Rothacher, M., Barthelmes, F., Neumayer, H., Biancale, R., Bruinsma, S., Lemoine, J.M., Loyer, S., “A Mean Global Gravity Field Model From the Combination of Satellite Mission and Altimetry/Gravimetry Surface Data - EIGEN-GL04C”, *Geophysical Research Abstracts*, Vol. 8, 03462, 2006.

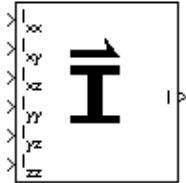
## Purpose

Create inertia tensor from moments and products of inertia

## Library

Mass Properties

## Description

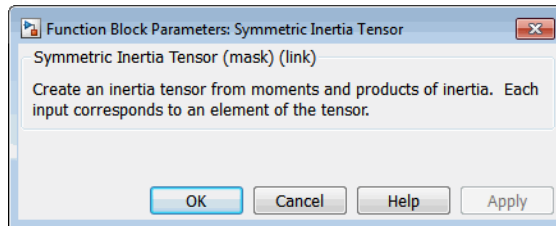


The Symmetric Inertia Tensor block creates an inertia tensor from moments and products of inertia. Each input corresponds to an element of the tensor.

The inertia tensor has the form of

$$Inertia = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{yz} \\ -I_{xy} & I_{yy} & -I_{xz} \\ -I_{yz} & -I_{xz} & I_{zz} \end{bmatrix}$$

## Dialog Box



## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the moment of inertia about the $x$ -axis.
Second		Contains the product of inertia in the $xy$ plane.
Third		Contains the product of inertia in the $xz$ plane.
Fourth		Contains the moment of inertia about the $y$ -axis.

# Symmetric Inertia Tensor

---

Input	Dimension Type	Description
Fifth		Contains the product of inertia in the $yz$ plane.
Sixth		Contains the moment of inertia about the $z$ -axis.

Output	Dimension Type	Description
First		Contains a symmetric 3-by-3 inertia tensor.

## See Also

Create 3x3 Matrix

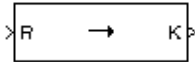
## Purpose

Convert from temperature units to desired temperature units

## Library

Utilities/Unit Conversions

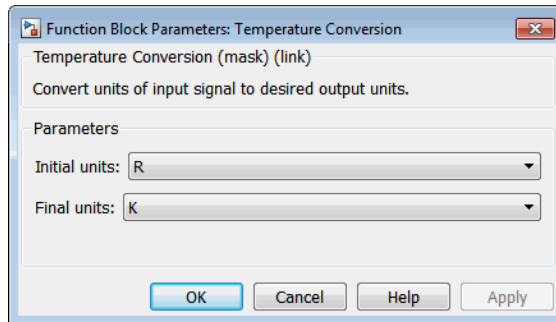
## Description



The Temperature Conversion block computes the conversion factor from specified input temperature units to specified output temperature units and applies the conversion factor to the input signal.

The Temperature Conversion block icon displays the input and output units selected from the **Initial units** and the **Final units** lists.

## Dialog Box



### Initial units

Specifies the input units.

### Final units

Specifies the output units.

The following conversion units are available:

K	Kelvin
F	Degrees Fahrenheit
C	Degrees Celsius
R	Degrees Rankine

# Temperature Conversion

---

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the temperature in initial temperature units.

Output	Dimension Type	Description
First		Contains the temperature in final temperature units.

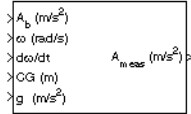
## See Also

Acceleration Conversion  
Angle Conversion  
Angular Acceleration Conversion  
Angular Velocity Conversion  
Density Conversion  
Force Conversion  
Length Conversion  
Mass Conversion  
Pressure Conversion  
Velocity Conversion

**Purpose** Implement three-axis accelerometer

**Library** GNC/Navigation

## Description



The Three-Axis Accelerometer block implements an accelerometer on each of the three axes. The ideal measured accelerations ( $\bar{A}_{imeas}$ ) include the acceleration in body axes at the center of gravity ( $\bar{A}_b$ ), lever arm effects due to the accelerometer not being at the center of gravity, and, optionally, gravity in body axes can be removed.

$$\bar{A}_{imeas} = \bar{A}_b + \bar{\omega}_b \times (\bar{\omega}_b \times \bar{d}) + \dot{\bar{\omega}}_b \times \bar{d} - \bar{g}$$

where  $\bar{\omega}_b$  are body-fixed angular rates,  $\dot{\bar{\omega}}_b$  are body-fixed angular accelerations and  $\bar{d}$  is the lever arm. The lever arm ( $\bar{d}$ ) is defined as the distances that the accelerometer group is forward, right and below the center of gravity.

$$\bar{d} = \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} = \begin{bmatrix} -(x_{acc} - x_{CG}) \\ y_{acc} - y_{CG} \\ -(z_{acc} - z_{CG}) \end{bmatrix}$$

The orientation of the axes used to determine the location of the accelerometer group ( $x_{acc}, y_{acc}, z_{acc}$ ) and center of gravity ( $x_{CG}, y_{CG}, z_{CG}$ ) is from the zero datum (typically the nose) to aft, to the right of the vertical centerline and above the horizontal centerline. The  $x$ -axis and  $z$ -axis of this measurement axes are opposite the body-fixed axes producing the negative signs in the lever arms for  $x$ -axis and  $z$ -axis.

Measured accelerations ( $\bar{A}_{meas}$ ) output by this block contain error sources and are defined as

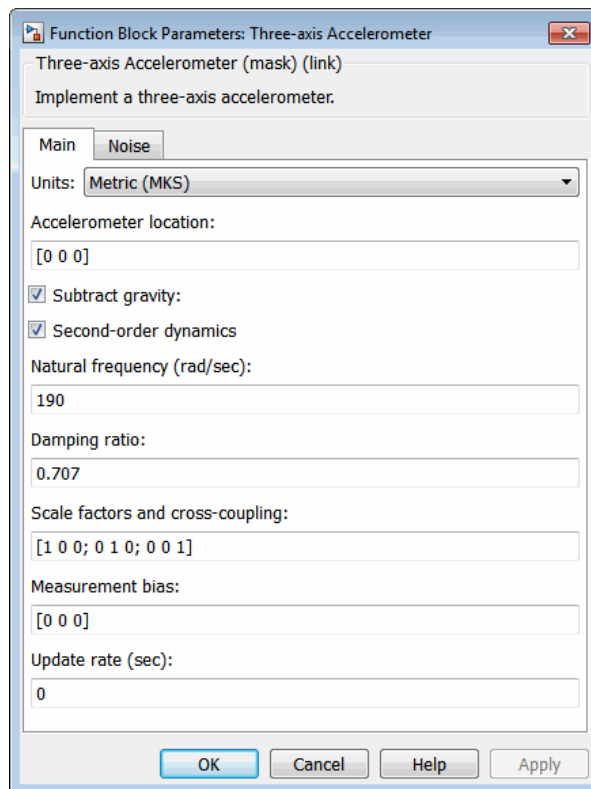
$$\bar{A}_{meas} = \bar{A}_{imeas} \times \bar{A}_{SFCC} + \bar{A}_{bias} + noise$$

# Three-Axis Accelerometer

where  $\bar{A}_{SFCC}$  is a 3-by-3 matrix of scaling factors on the diagonal and misalignment terms in the nondiagonal, and  $\bar{A}_{bias}$  are the biases.

Optionally discretizations can be applied to the block inputs and dynamics along with nonlinearizations of the measured accelerations via a Saturation block.

## Dialog Box

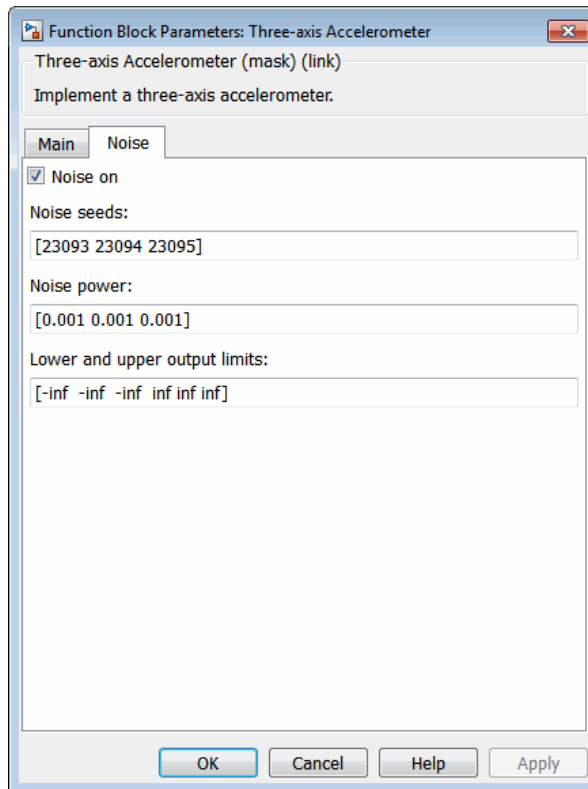


The dialog box, titled "Function Block Parameters: Three-axis Accelerometer", contains the following fields and controls:

- Three-axis Accelerometer (mask) (link)
- Implement a three-axis accelerometer.
- Tabbed interface with "Main" and "Noise" tabs.
- Units: Metric (MKS)
- Accelerometer location: [0 0 0]
- Subtract gravity:
- Second-order dynamics
- Natural frequency (rad/sec): 190
- Damping ratio: 0.707
- Scale factors and cross-coupling: [1 0 0; 0 1 0; 0 0 1]
- Measurement bias: [0 0 0]
- Update rate (sec): 0
- Buttons: OK, Cancel, Help, Apply



# Three-Axis Accelerometer



## Units

Specifies the input and output units:

<b>Units</b>	<b>Acceleration</b>	<b>Length</b>
Metric (MKS)	Meters per second squared	Meters
English	Feet per second squared	Feet

## Accelerometer location

The location of the accelerometer group is measured from the zero datum (typically the nose) to aft, to the right of the vertical

# Three-Axis Accelerometer

---

centerline and above the horizontal centerline. This measurement reference is the same for the center of gravity input. The units are in selected length units.

## **Subtract gravity**

Select to subtract gravity from acceleration readings.

## **Second order dynamics**

Select to apply second-order dynamics to acceleration readings.

## **Natural frequency (rad/sec)**

The natural frequency of the accelerometer. The units of natural frequency are radians per second.

## **Damping ratio**

The damping ratio of the accelerometer. A dimensionless parameter.

## **Scale factors and cross-coupling**

The 3-by-3 matrix used to skew the accelerometer from body axes and to scale accelerations along body axes.

## **Measurement bias**

The three-element vector containing long-term biases along the accelerometer axes. The units are in selected acceleration units.

## **Update rate (sec)**

Specify the update rate of the accelerometer. An update rate of 0 will create a continuous accelerometer. If noise is selected and the update rate is 0, then the noise will be updated at the rate of 0.1. The units of update rate are seconds.

## **Noise on**

Select to apply white noise to acceleration readings.

## **Noise seeds**

The scalar seeds for the Gaussian noise generator for each axis of the accelerometer.

## **Noise power**

The height of the PSD of the white noise for each axis of the accelerometer.

## Lower and upper output limits

The six-element vector containing three minimum values and three maximum values of acceleration in each of the accelerometer axes. The units are in selected acceleration units.

## Inputs and Outputs

Input	Dimension Type	Description
First	Three-element vector	Contains the actual accelerations in body-fixed axes, in selected units.
Second	Three-element vector	Contains the angular rates in body-fixed axes, in radians per second.
Third	Three-element vector	Contains the angular accelerations in body-fixed axes, in radians per second squared.
Fourth	Three-element vector	Contains the location of the center of gravity, in selected units.
Fifth (Optional)	Three-element vector	Contains the gravity, in selected units.

Output	Dimension Type	Description
First	Three-element vector	Contains the measured accelerations from the accelerometer, in selected units.

## Assumptions and Limitations

Vibropendulous error and hysteresis effects are not accounted for in this block. Additionally, this block is not intended to model the internal dynamics of different forms of the instrument.

**Note** This block requires the Control System Toolbox product for discrete operation (nonzero sample time).

# Three-Axis Accelerometer

---

## Reference

Rogers, R. M., *Applied Mathematics in Integrated Navigation Systems*, AIAA Education Series, 2000.

## See Also

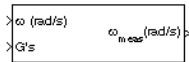
Three-Axis Gyroscope

Three-Axis Inertial Measurement Unit

**Purpose** Implement three-axis gyroscope

**Library** GNC/Navigation

**Description**



The Three-Axis Gyroscope block implements a gyroscope on each of the three axes. The measured body angular rates ( $\bar{\omega}_{meas}$ ) include the body angular rates ( $\bar{\omega}_b$ ), errors, and optionally discretizations and nonlinearizations of the signals.

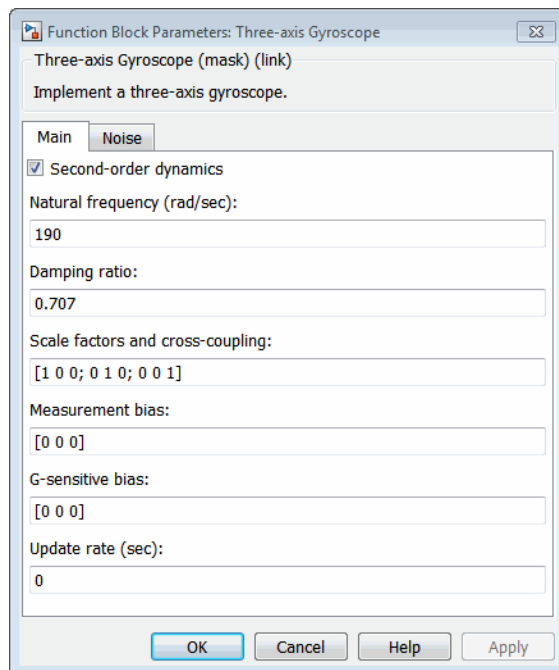
$$\bar{\omega}_{meas} = \bar{\omega}_b \times \bar{\omega}_{SFCC} + \bar{\omega}_{bias} + Gs \times \bar{\omega}_{gsens} + noise$$

where  $\bar{\omega}_{SFCC}$  is a 3-by-3 matrix of scaling factors on the diagonal and misalignment terms in the nondiagonal,  $\bar{\omega}_{bias}$  are the biases, ( $Gs$ ) are the  $Gs$  on the gyroscope, and  $\bar{\omega}_{gsens}$  are the g-sensitive biases.

Optionally discretizations can be applied to the block inputs and dynamics along with nonlinearizations of the measured body angular rates via a Saturation block.

# Three-Axis Gyroscope

## Dialog Box



Function Block Parameters: Three-axis Gyroscope

Three-axis Gyroscope (mask) (link)  
Implement a three-axis gyroscope.

Main Noise

Second-order dynamics

Natural frequency (rad/sec):  
190

Damping ratio:  
0.707

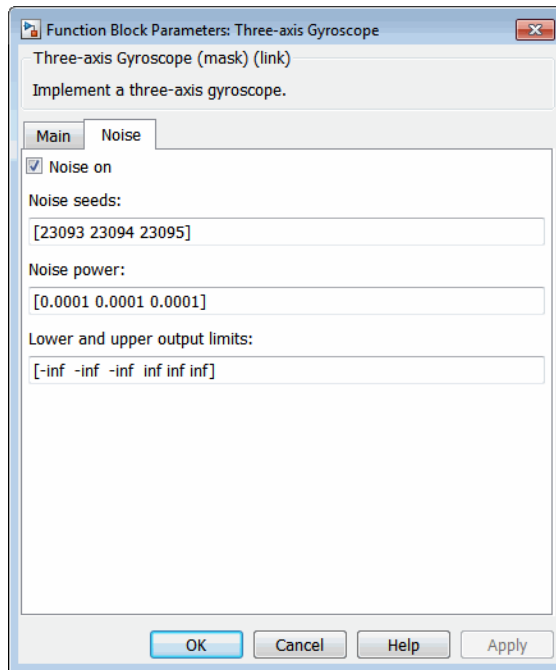
Scale factors and cross-coupling:  
[1 0 0; 0 1 0; 0 0 1]

Measurement bias:  
[0 0 0]

G-sensitive bias:  
[0 0 0]

Update rate (sec):  
0

OK Cancel Help Apply



## Second order dynamics

Select to apply second-order dynamics to gyroscope readings.

## Natural frequency (rad/sec)

The natural frequency of the gyroscope. The units of natural frequency are radians per second.

## Damping ratio

The damping ratio of the gyroscope. A dimensionless parameter.

## Scale factors and cross-coupling

The 3-by-3 matrix used to skew the gyroscope from body axes and to scale angular rates along body axes.

## Measurement bias

The three-element vector containing long-term biases along the gyroscope axes. The units are in radians per second.

# Three-Axis Gyroscope

---

## **G-sensitive bias**

The three-element vector contains the maximum change in rates due to linear acceleration. The units are in radians per second per g-unit.

## **Update rate (sec)**

Specify the update rate of the gyroscope. An update rate of 0 will create a continuous gyroscope. If noise is selected and the update rate is 0, then the noise will be updated at the rate of 0.1. The units of update rate are seconds.

## **Noise on**

Select to apply white noise to gyroscope readings.

## **Noise seeds**

The scalar seeds for the Gaussian noise generator for each axis of the gyroscope.

## **Noise power**

The height of the PSD of the white noise for each axis of the gyroscope.

## **Lower and upper output limits**

The six-element vector containing three minimum values and three maximum values of angular rates in each of the gyroscope axes. The units are in radians per second.

## **Inputs and Outputs**

<b>Input</b>	<b>Dimension Type</b>	<b>Description</b>
First	Three-element vector	Contains the angular rates in body-fixed axes, in radians per second.
Second	Three-element vector	Contains the accelerations in body-fixed axes, in Gs.



Output	Dimension Type	Description
First	Three-element vector	Contains the measured angular rates from the gyroscope, in radians per second.

## Assumptions and Limitations

Anisoelastic bias and anisoinertial bias effects are not accounted for in this block. Additionally, this block is not intended to model the internal dynamics of different forms of the instrument.

---

**Note** This block requires the Control System Toolbox product for discrete operation (nonzero sample time).

---

## Reference

Rogers, R. M., *Applied Mathematics in Integrated Navigation Systems*, AIAA Education Series, 2000.

## See Also

Three-Axis Accelerometer

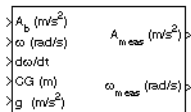
Three-Axis Inertial Measurement Unit

# Three-Axis Inertial Measurement Unit

**Purpose** Implement three-axis inertial measurement unit (IMU)

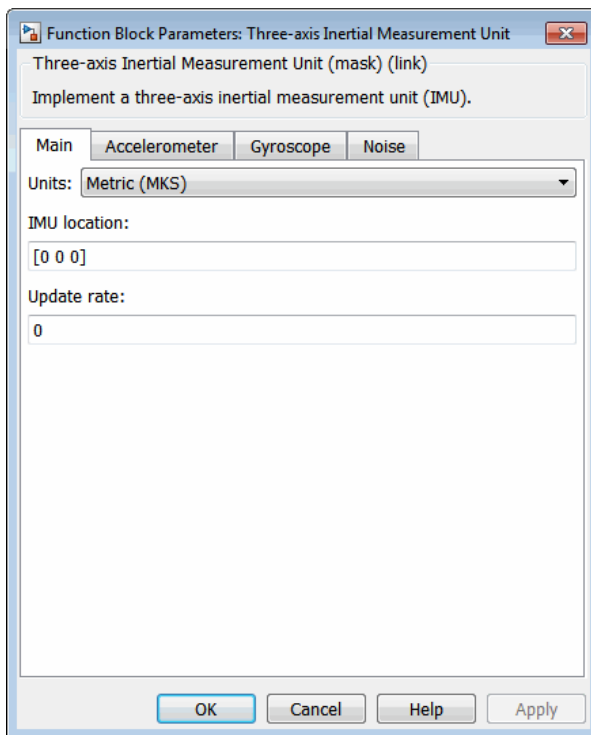
**Library** GNC/Navigation

**Description** The Three-Axis Inertial Measurement Unit block implements an inertial measurement unit (IMU) containing a three-axis accelerometer and a three-axis gyroscope.

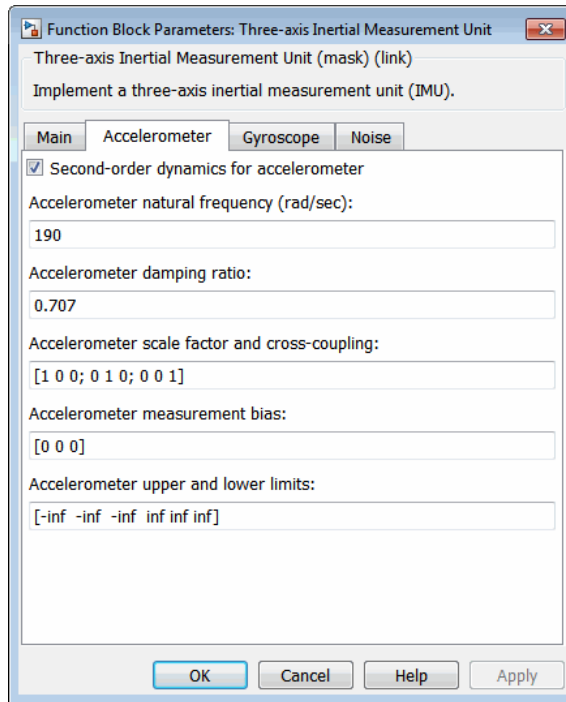


For a description of the equations and application of errors, see the Three-Axis Accelerometer block and the Three-Axis Gyroscope block reference pages.

## Dialog Box



# Three-Axis Inertial Measurement Unit



# Three-Axis Inertial Measurement Unit

Function Block Parameters: Three-axis Inertial Measurement Unit

Three-axis Inertial Measurement Unit (mask) (link)

Implement a three-axis inertial measurement unit (IMU).

Main Accelerometer Gyroscope Noise

Second-order dynamics for gyro

Gyro natural frequency (rad/sec):  
190

Gyro damping ratio:  
0.707

Gyro scale factors and cross-coupling:  
[1 0 0; 0 1 0; 0 0 1]

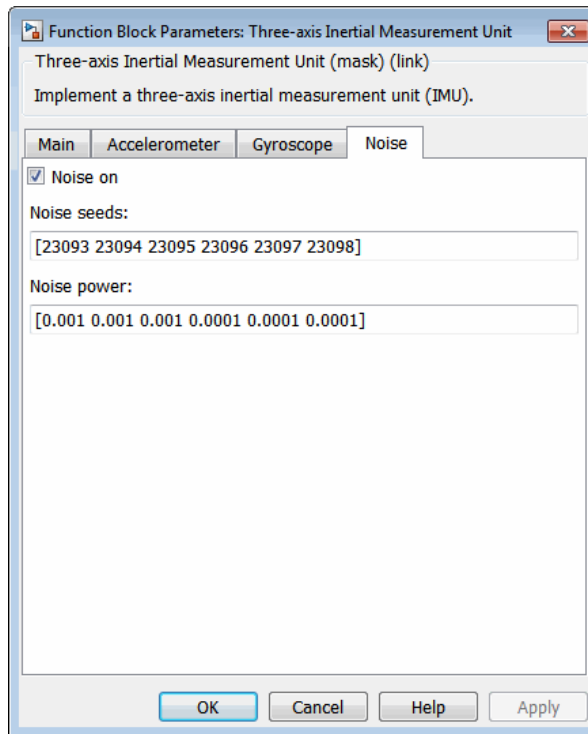
Gyro measurement bias:  
[0 0 0]

G-sensitive bias:  
[0 0 0]

Gyro upper and lower limits:  
[-inf -inf -inf inf inf inf]

OK Cancel Help Apply

# Three-Axis Inertial Measurement Unit



## Units

Specifies the input and output units:

Units	Acceleration	Length
Metric (MKS)	Meters per second squared	Meters
English	Feet per second squared	Feet

## IMU location

The location of the IMU, which is also the accelerometer group location, is measured from the zero datum (typically the nose) to aft, to the right of the vertical centerline and above the horizontal

# Three-Axis Inertial Measurement Unit

---

centerline. This measurement reference is the same for the center of gravity input. The units are in selected length units.

## **Update rate (sec)**

Specify the update rate of the accelerometer and gyroscope. An update rate of 0 will create a continuous accelerometer and continuous gyroscope. If noise is selected and the update rate is 0, then the noise will be updated at the rate of 0.1. The units of update rate are seconds.

## **Second order dynamics for accelerometer**

Select to apply second-order dynamics to acceleration readings.

## **Accelerometer natural frequency (rad/sec)**

The natural frequency of the accelerometer. The units of natural frequency are radians per second.

## **Accelerometer damping ratio**

The damping ratio of the accelerometer. A dimensionless parameter.

## **Accelerometer scale factors and cross-coupling**

The 3-by-3 matrix used to skew the accelerometer from body-axis and to scale accelerations along body-axis.

## **Accelerometer measurement bias**

The three-element vector containing long-term biases along the accelerometer axes. The units are in selected acceleration units.

## **Accelerometer lower and upper output limits**

The six-element vector containing three minimum values and three maximum values of acceleration in each of the accelerometer axes. The units are in selected acceleration units.

## **Gyro second order dynamics**

Select to apply second-order dynamics to gyroscope readings.

## **Gyro natural frequency (rad/sec)**

The natural frequency of the gyroscope. The units of natural frequency are radians per second.

# Three-Axis Inertial Measurement Unit

## **Gyro damping ratio**

The damping ratio of the gyroscope. A dimensionless parameter.

## **Gyro scale factors and cross-coupling**

The 3-by-3 matrix used to skew the gyroscope from body axes and to scale angular rates along body axes.

## **Gyro measurement bias**

The three-element vector containing long-term biases along the gyroscope axes. The units are in radians per second.

## **G-sensitive bias**

The three-element vector contains the maximum change in rates due to linear acceleration. The units are in radians per second per g-unit.

## **Gyro lower and upper output limits**

The six-element vector containing three minimum values and three maximum values of angular rates in each of the gyroscope axes. The units are in radians per second.

## **Noise on**

Select to apply white noise to acceleration and gyroscope readings.

## **Noise seeds**

The scalar seeds for the Gaussian noise generator for each axis of the accelerometer and gyroscope.

## **Noise power**

The height of the PSD of the white noise for each axis of the accelerometer and gyroscope.

## **Inputs and Outputs**

<b>Input</b>	<b>Dimension Type</b>	<b>Description</b>
First	Three-element vector	Contains the actual accelerations in body-fixed axes, in selected units.
Second	Three-element vector	Contains the angular rates in body-fixed axes, in radians per second.

# Three-Axis Inertial Measurement Unit

---

Input	Dimension Type	Description
Third	Three-element vector	Contains the angular accelerations in body-fixed axes, in radians per second squared.
Fourth	Three-element vector	Contains the location of the center of gravity, in selected units.
Fifth	Three-element vector	Contains the gravity, in selected units.

Output	Dimension Type	Description
First	Three-element vector	Contains the measured accelerations from the accelerometer, in selected units.
Second	Three-element vector	Contains the measured angular rates from the gyroscope, in radians per second.

## Assumptions and Limitations

Vibropendulous error, hysteresis affects, anisoelastic bias and anisoinertial bias are not accounted for in this block. Additionally, this block is not intended to model the internal dynamics of different forms of the instrument.

---

**Note** This block requires the Control System Toolbox product for discrete operation (nonzero sample time).

---

## Examples

See asbh120 for an example of this block.

## Reference

Rogers, R. M., *Applied Mathematics in Integrated Navigation Systems*, AIAA Education Series, 2000.



# Three-Axis Inertial Measurement Unit

---

## **See Also**

Three-Axis Accelerometer

Three-Axis Gyroscope

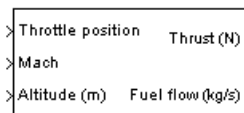
# Turbofan Engine System

---

**Purpose** Implement first-order representation of turbofan engine with controller

**Library** Propulsion

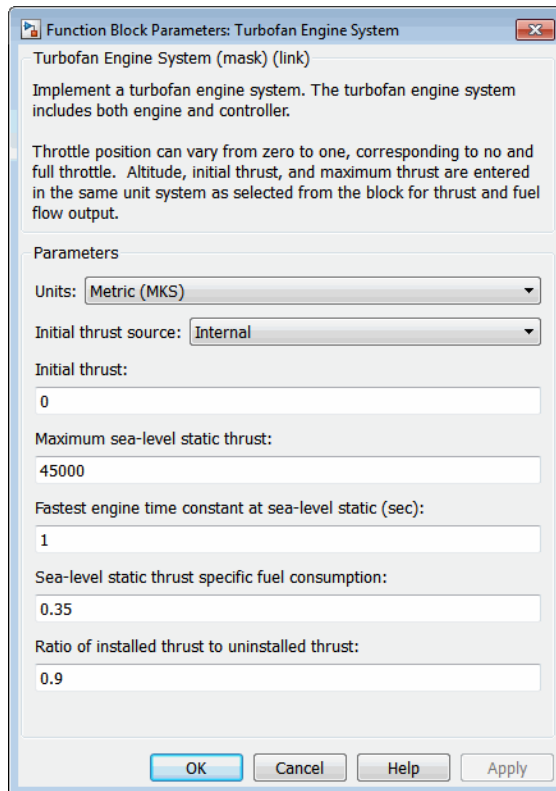
**Description** The Turbofan Engine System block computes the thrust and the weight of fuel flow of a turbofan engine and controller at a specific throttle position, Mach number, and altitude.



This system is represented by a first-order system with unitless heuristic lookup tables for thrust, thrust specific fuel consumption (TSFC), and engine time constant. For the lookup table data, thrust is a function of throttle position and Mach number, TSFC is a function of thrust and Mach number, and engine time constant is a function of thrust. The unitless lookup table outputs are corrected for altitude using the relative pressure ratio  $\delta$  and relative temperature ratio  $\theta$ , and scaled by maximum sea level static thrust, fastest engine time constant at sea level static, sea level static thrust specific fuel consumption, and ratio of installed thrust to uninstalled thrust.

The Turbofan Engine System block icon displays the input and output units selected from the **Units** list.

## Dialog Box



### Units

Specifies the input and output units:

<b>Units</b>	<b>Altitude</b>	<b>Thrust</b>	<b>Fuel Flow</b>
Metric (MKS)	Meters	Newtons	Kilograms per second
English	Feet	Pound force	Pound mass per second

# Turbofan Engine System

---

## Initial thrust source

Specifies the source of initial thrust:

Internal	Use initial thrust value from mask dialog.
External	Use external input for initial thrust value.

## Initial thrust

Initial value for thrust.

## Maximum sea-level static thrust

Maximum thrust at sea-level and at Mach = 0.

## Fastest engine time constant at sea-level static

Fastest engine time at sea level.

## Sea-level static thrust specific fuel consumption

Thrust specific fuel consumption at sea level, at Mach = 0, and at maximum thrust, in specified mass units per hour per specified thrust units.

## Ratio of installed thrust to uninstalled thrust

Coefficient representing the loss in thrust due to engine installation.

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the throttle position, which can vary from zero to one, corresponding to no and full throttle.
Second		Contains the Mach number.
Third		Contains the altitude in specified length units.

Output	Dimension Type	Description
First		Contains the thrust in specified force units.
Second		Contains the fuel flow in specified mass units per second.

## Assumptions and Limitations

The atmosphere is at standard day conditions and an ideal gas.

The Mach number is limited to less than 1.0.

This engine system is for indication purposes only. It is not meant to be used as a reference model.

This engine system is assumed to have a high bypass ratio.

## References

*Aeronautical Vestpocket Handbook*, United Technologies Pratt & Whitney, August, 1986.

Raymer, D. P., *Aircraft Design: A Conceptual Approach*, AIAA Education Series, Washington, DC, 1989.

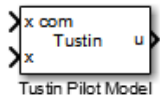
Hill, P. G., and C. R. Peterson, *Mechanics and Thermodynamics of Propulsion*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1970.

# Tustin Pilot Model

**Purpose** Represent Tustin pilot model

**Library** Pilot Models

## Description



The Tustin Pilot Model block represents the pilot model that A. Tustin describes in *The Nature of the Operator's Response in Manual Control, and its Implications for Controller Design*. (For more information, see [1].) When modeling human pilot models, use this block for the least accuracy, compared to that provided by the Crossover Pilot Model and Precision Pilot Model blocks. This block requires less input than those blocks, and provides better performance. However, the results might be less accurate.

This pilot model is a single input, single output (SISO) model that represents human behavior, based on the transfer function:

$$\frac{u(s)}{e(s)} = \frac{K_p(1 + Ts)}{s} e^{-\tau s}.$$

In this equation:

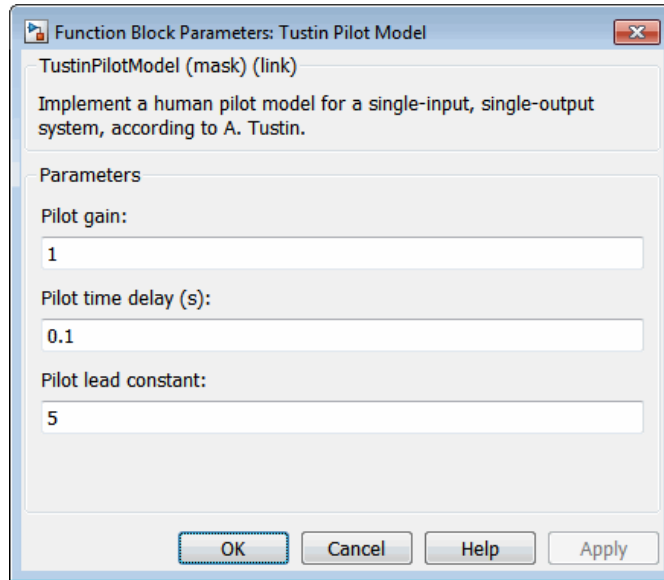
Variable	Description
$K_p$	Pilot gain.
$T$	Lead constant.
$\tau$	Transport delay time caused by the pilot neuromuscular system.
$u(s)$	Input to the aircraft model and output to the pilot model.
$e(s)$	Error between the desired pilot value and the actual value.

This block has non-linear behavior. If you want to linearize the block (for example, with one of the Simulink `linmod` functions), you might need to change the Pade approximation order. The Tustin Pilot Model block implementation incorporates the Simulink Transport Delay block

with the **Pade order (for linearization)** parameter set to 2 by default. To change this value, use the `set_param` function, for example:

```
set_param(gcb, 'pade', '3')
```

## Dialog Box



### Pilot gain

Specifies the pilot gain.

### Pilot time delay (s)

Specifies the total pilot time delay, in seconds. This value typically ranges from 0.1 s to 0.2 s.

### Pilot lead constant

Specifies the pilot lead constant.

# Tustin Pilot Model

---

## Inputs and Outputs

Input	Dimension Type	Description
First	1-by-1	Contains the command for the signal that the pilot model controls.
Second	1-by-1	Contains the signal that the pilot model controls.

Output	Dimension Type	Description
First	1-by-1	Contains the command for the aircraft.

## References

[1] Tustin, A., *The Nature of the Operator's Response in Manual Control, and its Implications for Controller Design*. Convention on Automatic Regulators and Servo Mechanisms. May, 1947.

## See Also

Crossover Pilot Model | Precision Pilot Model |



# Unpack net\_ctrl Packet from FlightGear

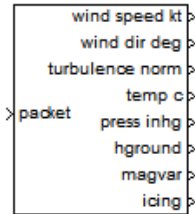
## Purpose

Unpack net\_ctrl variable packet received from FlightGear

## Library

Animation/Flight Simulator Interfaces

## Description



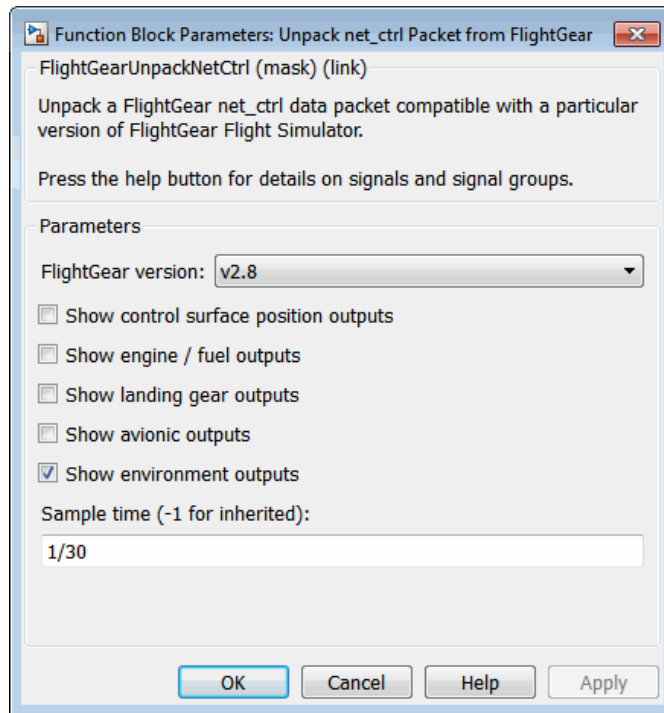
The Unpack net\_ctrl Packet from FlightGear block unpacks net\_ctrl variable packets received from FlightGear and makes them available for the Simulink environment.

Supported FlightGear versions:

- v0.9.3
- v0.9.8/0.9.8a
- v0.9.9
- v0.9.10
- v1.0
- v1.9.1
- v2.0
- v2.4
- v2.6
- v2.8

# Unpack net\_ctrl Packet from FlightGear

## Dialog Box



### FlightGear version

Select your FlightGear software version.

Supported FlightGear versions: v0.9.3, v0.9.8/0.9.8a, v0.9.9, v0.9.10, v1.0, v1.9.1, v2.0, v2.4, v2.6, v2.8.

### Show control surface position outputs

Select this check box to include the control surface position outputs (signal group 1) from the FlightGear net\_ctrl data packet.

### Show engine/fuel outputs

Select this check box to include the engine and fuel outputs (signal group 2) from the FlightGear net\_ctrl data packet.

# Unpack net\_ctrl Packet from FlightGear

## Show landing gear outputs

Select this check box to include the landing gear outputs (signal group 3) from the FlightGear net\_ctrl data packet.

## Show avionic outputs

Select this check box to include the avionic outputs (signal group 4) from the FlightGear net\_ctrl data packet.

## Show environment outputs

Select this check box to include the environment outputs (signal group 5) from the FlightGear net\_ctrl data packet.

## Sample time

Specify the sample time (-1 for inherited).

## Inputs and Outputs

## Output Signals Supported for FlightGear 0.9.10/1.0/1.9.1/2.0/2.4/2.6/2.8

This table lists all the output signals supported for Versions 0.9.10, 1.0, 1.9.1, 2.0, 2.4, 2.6, and 2.8:

### Signal Group 1: Control surface position outputs

Name	Units	Type	Width	Description
<i>aileron</i>	geometry specific units [-1,1]	double	1	Aileron position.
<i>elevator</i>	geometry specific units [-1,1]	double	1	Elevator position.
<i>rudder</i>	geometry specific units [-1,1]	double	1	Rudder position.
<i>aileron_trim</i>	geometry specific units [-1,1]	double	1	Aileron trim position.

# Unpack net\_ctrl Packet from FlightGear

## Signal Group 1: Control surface position outputs (Continued)

Name	Units	Type	Width	Description
<i>elevator_trim</i>	geometry specific units [-1,1]	double	1	Elevator trim position.
<i>rudder_trim</i>	geometry specific units [0,1]	double	1	Rudder trim position.
<i>flaps</i>	geometry specific units	double	1	Flaps position.
<i>spoilers</i>	geometry specific units	double	1	Spoilers position.
<i>flaps_power</i>	—	single	1	Power administered to the flaps (1 = power available).
<i>flap_motor_ok</i>	—	single	1	Shows whether or not the flap motor is working properly.

## Signal Group 2: Engine/fuel outputs

Name	Units	Type	Width	Description
<i>num_engines</i>	—	single	1	Number of valid engines.
<i>master_bat</i>	—	single	1	Master battery switch position.
<i>master_alt</i>	—	single	1	Master alternator switch position.
<i>magnetos</i>	—	single	1	Magnetos switch position.

# Unpack net\_ctrl Packet from FlightGear

## Signal Group 2: Engine/fuel outputs (Continued)

Name	Units	Type	Width	Description
<i>starter_power</i>	geometry specific units [0,1]	single	1	Starter power switch position (1 = starter power).
<i>throttle</i>	geometry specific units [0,1]	double	1	Throttle position.
<i>mixture</i>	geometry specific units [0,1]	double	1	Fuel mixture lever position.
<i>condition</i>	—	double	1	Condition lever position.
<i>fuel_pump_power</i>	—	single	1	Fuel pump power switch position (1 = on).
<i>prop_advance</i>	—	double	1	Propeller blade advance ratio.
<i>feed_tank_to</i>	—	single	1	Tank feed switch position.
<i>reverse</i>	—	single	1	Reverse switch position.
<i>engine_ok</i>	—	single	1	Indicates whether the engine is okay or not.
<i>mag_left_ok</i>	—	single	1	Indicates whether the left magneto is okay or not.

# Unpack net\_ctrl Packet from FlightGear

## Signal Group 2: Engine/fuel outputs (Continued)

Name	Units	Type	Width	Description
<i>mag_right_ok</i>	—	single	1	Indicates whether the right magneto is okay or not.
<i>spark_plugs_ok</i>	—	single		Indicates whether the spark plugs are fouled or not (0 = fouled plugs).
<i>oil_press_status</i>	—	single		Oil pressure status indicator (0 = normal, 1 = low, 2 = full failure).
<i>fuel_pump_ok</i>	—	single		Indicates fuel pump status.
<i>num_tanks</i>	—	single		Number of valid tanks.
<i>fuel_selector</i>	—	single		Fuel selector switch position. There is one per each fuel tank. (0 = off, 1 = on).
<i>xfer_pump</i>	—	single		Specifies transfer from array value to tank specified by an integer value.
<i>cross_feed</i>	—	single		Cross feed fuel pump switch indicator (0 = off, 1 = on).

# Unpack net\_ctrl Packet from FlightGear

## Signal Group 3: Landing gear outputs

Name	Units	Type	Width	Description
<i>brake_left</i>	geometry-specific units	double	1	Left brake position for the pilot.
<i>brake_right</i>	geometry-specific units	double	1	Right brake position for the pilot.
<i>copilot_brake_left</i>	geometry-specific units	double	1	Left brake position for the copilot.
<i>copilot_brake_right</i>	geometry-specific units	double	1	Right brake position for the copilot.
<i>brake_parking</i>	geometry-specific units	double	1	Parking brake position.
<i>gear_handle</i>	—	single	1	Landing gear handle position (0 = gear handle up).

## Signal Group 4: Avionics outputs

Name	Units	Type	Width	Description
<i>master_avionics</i>	—	single	1	Master avionics switch.
<i>comm_1</i>	Hz	double	1	Communications primary frequency.
<i>comm_2</i>	Hz	double	1	Communications secondary frequency.
<i>nav_1</i>	Hz	double	1	Navigation primary frequency.
<i>nav_2</i>	Hz	double	1	Navigation secondary frequency.

# Unpack net\_ctrl Packet from FlightGear

## Signal Group 5: Environment outputs

Name	Units	Type	Width	Description
<i>wind_speed_kt</i>	knots	double	1	Wind speed.
<i>wind_dir_deg</i>	deg	double	1	Wind direction.
<i>turbulence_norm</i>	—	double	1	Turbulence magnitude, normalized between 0 and 1.
<i>temp_c</i>	Degrees Celsius	double	1	Outside temperature.
<i>press_inhg</i>	in Hg	double	1	Outside pressure.
<i>hground</i>	meters	double	1	Ground elevation.
<i>magvar</i>	deg	double	1	Local magnetic variation.
<i>icing</i>	—	single	1	Icing conditions.

## Signal Group 6: Show Environment Inputs

Name	Units	Type	Width	Description
<i>agl</i>	m	single	1	Above ground level.
<i>cur_time</i>	sec	int32	1	Current UNIX time.
<i>warp</i>	sec	int32	1	Offset in seconds to UNIX time.
<i>visibility</i>	m	single	1	Visibility in meters (for visual effects).

## Output Signals Supported for FlightGear 0.9.9

This table lists all the output signals supported for Version 0.9.9:



# Unpack net\_ctrl Packet from FlightGear

## Signal Group 1: Surface position outputs

Name	Units	Type	Width	Description
<i>aileron</i>	geometry specific units [-1,1]	double	1	Aileron position.
<i>elevator</i>	geometry specific units [-1,1]	double	1	Elevator position.
<i>rudder</i>	geometry specific units [-1,1]	double	1	Rudder position.
<i>aileron_trim</i>	geometry specific units [-1,1]	double	1	Aileron trim position.
<i>elevator_trim</i>	geometry specific units [-1,1]	double	1	Elevator trim position.
<i>rudder_trim</i>	geometry specific units [-1,1]	double	1	Rudder trim position.
<i>flaps</i>	geometry specific units [0,1]	double	1	Flaps position.
<i>flaps_power</i>	—	single	1	Power administered to the flaps (1 = power available).
<i>flap_motor_ok</i>	—	single	1	Shows whether or not the flap motor is working properly.

# Unpack net\_ctrl Packet from FlightGear

## Signal Group 2: Engine/fuel outputs

Name	Units	Type	Width	Description
<i>num_engines</i>	—	single	1	Number of valid engines.
<i>master_bat</i>	—	single	4	Master battery switch position.
<i>master_alt</i>	—	single	4	Master alternator switch position.
<i>magnetos</i>	—	single	4	Magnetos switch position.
<i>starter_power</i>	—	single	4	Starter power switch position (1 = starter power).
<i>throttle</i>	geometry specific units [0,1]	double	4	Throttle position.
<i>mixture</i>	geometry specific units [0,1]	double	4	Fuel mixture lever position.
<i>condition</i>	geometry specific units [0,1]	double	4	Condition lever position.
<i>fuel_pump_power</i>	—	single	4	Fuel pump power switch position (1 = on).
<i>prop_advance</i>	—	double	4	Propeller blade advance ratio.
<i>engine_ok</i>	—	single	4	Indicates whether the engine is okay or not.

# Unpack net\_ctrl Packet from FlightGear

## Signal Group 2: Engine/fuel outputs (Continued)

Name	Units	Type	Width	Description
<i>mag_left_ok</i>	—	single	4	Indicates whether the left magneto is okay or not.
<i>mag_right_ok</i>	—	single	4	Indicates whether the right magneto is okay or not.
<i>spark_plugs_ok</i>	—	single	4	Indicates whether the spark plugs are fouled or not (0 = fouled plugs).
<i>oil_press_status</i>	—	single	4	Oil pressure status indicator (0 = normal, 1 = low, 2 = full failure).
<i>fuel_pump_ok</i>	—	single	4	Indicates fuel pump status.
<i>num_tanks</i>	—	single	1	Number of valid tanks.
<i>fuel_selector</i>	—	single	8	Fuel selector switch position. There is one per each fuel tank. (0 = off, 1 = on).
<i>cross_feed</i>	—	single	1	Cross feed fuel pump switch indicator (0 = off, 1 = on).

# Unpack net\_ctrl Packet from FlightGear

## Signal Group 3: Landing gear outputs

Name	Units	Type	Width	Description
<i>brake_left</i>	geometry-specific units	double	1	Left brake position for the pilot.
<i>brake_right</i>	geometry-specific units	double	1	Right brake position for the pilot.
<i>copilot_brake_left</i>	geometry-specific units	double	1	Left brake position for the copilot.
<i>copilot_brake_right</i>	geometry-specific units	double	1	Right brake position for the copilot.
<i>brake_parking</i>	geometry-specific units	double	1	Parking brake position.
<i>gear_handle</i>	—	single	1	Landing gear handle position (0 = gear handle up).

## Signal Group 4: Avionics outputs

Name	Units	Type	Width	Description
<i>master_avionics</i>	—	single	1	Master avionics switch.

## Signal Group 5: Environment outputs

Name	Units	Type	Width	Description
<i>wind_speed_kt</i>	knots	double	1	Wind speed.
<i>wind_dir_deg</i>	deg	double	1	Wind direction.

# Unpack net\_ctrl Packet from FlightGear

## Signal Group 5: Environment outputs (Continued)

Name	Units	Type	Width	Description
<i>turbulence_norm</i>	—	double	1	Turbulence magnitude, normalized between 0 and 1.
<i>temp_c</i>	Degrees Celsius	double	1	Outside temperature
<i>press_inhg</i>	In Hg	double	1	Outside pressure.
<i>hground</i>	meters	double	1	Ground elevation.
<i>magvar</i>	deg	double	1	Local magnetic variation.
<i>icing</i>	—	single	1	Icing conditions.

## Output Signals Supported for FlightGear 0.9.8

This table lists all the output signals supported for Version 0.9.8.

## Signal Group 1: Surface position outputs

Name	Units	Type	Width	Description
<i>aileron</i>	geometry specific units [-1,1]	double	1	Aileron position.
<i>elevator</i>	geometry specific units [-1,1]	double	1	Elevator position.
<i>elevator_trim</i>	geometry specific units [-1,1]	double	1	Elevator trim position.

# Unpack net\_ctrl Packet from FlightGear

## Signal Group 1: Surface position outputs (Continued)

Name	Units	Type	Width	Description
<i>rudder</i>	geometry specific units [-1,1]	double	1	Rudder position.
<i>flaps</i>	geometry specific units [0,1]	double	1	Flaps position.
<i>flaps_power</i>	—	double	1	Power administered to the flaps (1 = power available).
<i>flap_motor_ok</i>	—	bool	1	Shows whether or not the flap motor is working properly.

## Signal Group 2: Engine/fuel outputs

Name	Units	Type	Width	Description
<i>num_engines</i>	—	int	1	Number of valid engines.
<i>master_bat</i>	—	bool	4	Master battery switch position.
<i>master_alt</i>	—	bool	4	Master alternator switch position.
<i>magnetos</i>	—	int	4	Magnetos switch position.
<i>starter_power</i>	—	bool	4	Starter power switch position (1 = starter power).

# Unpack net\_ctrl Packet from FlightGear

## Signal Group 2: Engine/fuel outputs (Continued)

Name	Units	Type	Width	Description
<i>throttle</i>	geometry specific units [0,1]	double	4	Throttle position.
<i>mixture</i>	geometry specific units [0,1]	double	4	Fuel mixture lever position.
<i>condition</i>	geometry specific units [0,1]	double	4	Condition lever position.
<i>fuel_pump_power</i>	—	bool	4	Fuel pump power switch position (1 = on).
<i>prop_advance</i>	—	double	4	Propeller blade advance ratio.
<i>engine_ok</i>	—	bool	4	Indicates whether the engine is okay or not.
<i>mag_left_ok</i>	—	bool	4	Indicates whether the left magneto is okay or not.
<i>mag_right_ok</i>	—	bool	4	Indicates whether the right magneto is okay or not.
<i>spark_plugs_ok</i>	—	bool	4	Indicates whether the spark plugs are fouled or not (0 = fouled plugs).

# Unpack net\_ctrl Packet from FlightGear

## Signal Group 2: Engine/fuel outputs (Continued)

Name	Units	Type	Width	Description
<i>oil_press_status</i>	—	int	4	Oil pressure status indicator (0 = normal, 1 = low, 2 = full failure)
<i>fuel_pump_ok</i>	—	bool	4	Indicates fuel pump status.
<i>num_tanks</i>	—	int	1	Number of valid tanks.
<i>fuel_selector</i>	—	bool	8	Fuel selector switch position. There is one per each fuel tank. (0 = off, 1 = on).

## Signal Group 3: Landing gear outputs

Name	Units	Type	Width	Description
<i>brake_left</i>	geometry-specific units	double	1	Left brake position for the pilot.
<i>brake_right</i>	geometry-specific units	double	1	Right brake position for the pilot.
<i>copilot_brake_left</i>	geometry-specific units	double	1	Left brake position for the pilot.
<i>copilot_brake_right</i>	geometry-specific units	double	1	Right brake position for the pilot.
<i>brake_parking</i>	geometry-specific units	double	1	Parking brake position.
<i>gear_handle</i>	—	bool	1	Landing gear handle position (0 = gear handle up).



# Unpack net\_ctrl Packet from FlightGear

## Signal Group 4: Avionics outputs

Name	Units	Type	Width	Description
<i>master_avionics</i>	—	bool	1	Master avionics switch.

## Signal Group 5: Environment outputs

Name	Units	Type	Width	Description
<i>wind_speed_kt</i>	knots	double	1	Wind speed.
<i>wind_dir_deg</i>	deg	double	1	Wind direction.
<i>turbulence_norm</i>	—	double	1	Turbulence magnitude, normalized between 0 and 1.
<i>temp_c</i>	Degrees Celsius	double	1	Outside temperature.
<i>press_inhg</i>	In Hg	double	1	Outside pressure.
<i>hground</i>	meters	double	1	Ground elevation.
<i>magvar</i>	deg	double	1	Local magnetic variation.
<i>icing</i>	—	bool	1	Icing conditions.

## Output Signals Supported for FlightGear 0.9.3

This table lists all the output signals supported for Version 0.9.3.

# Unpack net\_ctrl Packet from FlightGear

## Signal Group 1: Surface position outputs

Name	Units	Type	Width	Description
<i>aileron</i>	geometry specific units [-1,1]	double	1	Aileron position.
<i>elevator</i>	geometry specific units [-1,1]	double	1	Elevator position.
<i>elevator_trim</i>	geometry specific units [-1,1]	double	1	Elevator trim position.
<i>rudder</i>	geometry specific units [-1,1]	double	1	Rudder position.
<i>flaps</i>	geometry specific units [0,1]	double	1	Flaps position.
<i>flaps_power</i>	—	bool	1	Power administered to the flaps (1 = power available).
<i>flap_motor_ok</i>	—	bool	1	Shows whether or not the flap motor is working properly.

## Signal Group 2: Engine/fuel outputs

Name	Units	Type	Width	Description
<i>num_engines</i>	—	int	1	Number of valid engines.
<i>magnetos</i>	—	int	4	Magnetos switch position.

# Unpack net\_ctrl Packet from FlightGear

## Signal Group 2: Engine/fuel outputs (Continued)

Name	Units	Type	Width	Description
<i>starter_power</i>	—	bool	4	Starter power switch position (1 = starter power).
<i>throttle</i>	geometry specific units [0,1]	double	4	Throttle position.
<i>mixture</i>	geometry specific units [0,1]	double	4	Fuel mixture lever position.
<i>fuel_pump_power</i>	—	bool	4	Fuel pump power switch position (1 = on).
<i>prop_advance</i>	—	double	4	Propeller blade advance ratio.
<i>engine_ok</i>	—	bool	4	Indicates whether the engine is okay or not.
<i>mag_left_ok</i>	—	bool	4	Indicates whether the left magneto is okay or not.
<i>mag_right_ok</i>	—	bool	4	Indicates whether the right magneto is okay or not.
<i>spark_plugs_ok</i>	—	bool	4	Indicates whether the spark plugs are fouled or not (0 = fouled plugs).

# Unpack net\_ctrl Packet from FlightGear

## Signal Group 2: Engine/fuel outputs (Continued)

Name	Units	Type	Width	Description
<i>oil_press_status</i>	—	int	4	Oil pressure status indicator (0 = normal, 1 = low, 2 = full failure).
<i>fuel_pump_ok</i>	—	bool	4	Indicates fuel pump status.
<i>num_tanks</i>	—	int	1	Number of valid tanks.
<i>fuel_selector</i>	—	bool	8	Fuel selector switch position. There is one per each fuel tank. (0 = off, 1 = on).
<i>master_bat</i>	—	bool	1	Master battery switch position.
<i>master_alt</i>	—	bool	1	Master alternator switch position.

## Signal Group 3: Landing gear outputs

Num	Units	Type	Width	Description
<i>num_wheels</i>	—	int	1	Number of valid wheels.
<i>brake</i>	—	double	16	Brake position for each wheel [0,1].
<i>gear_handle</i>	—	bool	1	Landing gear handle position (0 = gear handle up).

# Unpack net\_ctrl Packet from FlightGear

## Signal Group 4: Avionics outputs

Name	Units	Type	Width	Description
<i>master_avionics</i>	—	bool	1	Master avionics switch.

## Signal Group 5: Environment outputs

Name	Units	Type	Width	Description
<i>wind_speed_kt</i>	knots	double	1	Wind speed.
<i>wind_dir_deg</i>	deg	double	1	Wind direction.
<i>turbulence_norm</i>	—	double	1	Turbulence magnitude, normalized between 0 and 1.
<i>hground</i>	meters	double	1	Ground elevation.
<i>magvar</i>	deg	double	1	Local magnetic variation.

**Examples** See asbh120.

**See Also** FlightGear Preconfigured 6DoF Animation  
Generate Run Script  
Pack net\_fdm Packet for FlightGear  
Receive net\_ctrl Packet from FlightGear

# Velocity Conversion

**Purpose** Convert from velocity units to desired velocity units

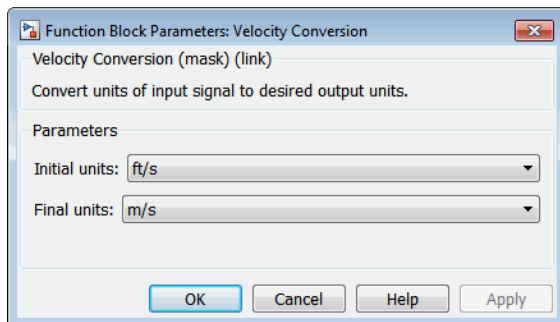
**Library** Utilities/Unit Conversions

**Description** The Velocity Conversion block computes the conversion factor from specified input velocity units to specified output velocity units and applies the conversion factor to the input signal.



The Velocity Conversion block icon displays the input and output units selected from the **Initial units** and the **Final units** lists.

## Dialog Box



**Initial units**  
Specifies the input units.

**Final units**  
Specifies the output units.

The following conversion units are available:

m/s	Meters per second
ft/s	Feet per second
km/s	Kilometers per second
in/s	Inches per second
km/h	Kilometers per hour

mph	Miles per hour
kts	Nautical miles per hour
ft/min	Feet per minute

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the velocity in initial velocity units.

Output	Dimension Type	Description
First		Contains the velocity in final velocity units.

## See Also

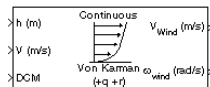
Acceleration Conversion  
Angle Conversion  
Angular Acceleration Conversion  
Angular Velocity Conversion  
Density Conversion  
Force Conversion  
Length Conversion  
Mass Conversion  
Pressure Conversion  
Temperature Conversion

# Von Karman Wind Turbulence Model (Continuous)

**Purpose** Generate continuous wind turbulence with Von Kármán velocity spectra

**Library** Environment/Wind

## Description



The Von Kármán Wind Turbulence Model (Continuous) block uses the Von Kármán spectral representation to add turbulence to the aerospace model by passing band-limited white noise through appropriate forming filters. This block implements the mathematical representation in the Military Specification MIL-F-8785C and Military Handbook MIL-HDBK-1797.

According to the military references, turbulence is a stochastic process defined by velocity spectra. For an aircraft flying at a speed  $V$  through a frozen turbulence field with a spatial frequency of  $\Omega$  radians per meter, the circular frequency  $\omega$  is calculated by multiplying  $V$  by  $\Omega$ . The following table displays the component spectra functions:

	MIL-F-8785C	MIL-HDBK-1797
<b>Longitudinal</b>		
$\Phi_u(\omega)$	$\frac{2\sigma_u^2 L_u}{\pi V} \cdot \frac{1}{\left[1 + \left(1.339 L_u \frac{\omega}{V}\right)^2\right]^{5/6}}$	$\frac{2\sigma_u^2 L_u}{\pi V} \cdot \frac{1}{\left[1 + \left(1.339 L_u \frac{\omega}{V}\right)^2\right]^{5/6}}$
$\Phi_p(\omega)$	$\frac{\sigma_w^2}{V L_w} \cdot \frac{0.8 \left(\frac{\pi L_w}{4b}\right)^{1/3}}{1 + \left(\frac{4b\omega}{\pi V}\right)^2}$	$\frac{\sigma_w^2}{2V L_w} \cdot \frac{0.8 \left(\frac{2\pi L_w}{4b}\right)^{1/3}}{1 + \left(\frac{4b\omega}{\pi V}\right)^2}$
<b>Lateral</b>		



# Von Karman Wind Turbulence Model (Continuous)

	<b>MIL-F-8785C</b>	<b>MIL-HDBK-1797</b>
$\Phi_v(\omega)$	$\frac{\sigma_v^2 L_v}{\pi V} \cdot \frac{1 + \frac{8}{3} \left( 1.339 L_v \frac{\omega}{V} \right)^2}{\left[ 1 + \left( 1.339 L_v \frac{\omega}{V} \right)^2 \right]^{11/6}}$	$\frac{2\sigma_v^2 L_v}{\pi V} \cdot \frac{1 + \frac{8}{3} \left( 2.678 L_v \frac{\omega}{V} \right)^2}{\left[ 1 + \left( 2.678 L_v \frac{\omega}{V} \right)^2 \right]^{11/6}}$
$\Phi_r(\omega)$	$\frac{\mp \left( \frac{\omega}{V} \right)^2}{1 + \left( \frac{3b\omega}{\pi V} \right)^2} \cdot \Phi_v(\omega)$	$\frac{\mp \left( \frac{\omega}{V} \right)^2}{1 + \left( \frac{3b\omega}{\pi V} \right)^2} \cdot \Phi_v(\omega)$
<b>Vertical</b>		
$\Phi_w(\omega)$	$\frac{\sigma_w^2 L_w}{\pi V} \cdot \frac{1 + \frac{8}{3} \left( 1.339 L_w \frac{\omega}{V} \right)^2}{\left[ 1 + \left( 1.339 L_w \frac{\omega}{V} \right)^2 \right]^{11/6}}$	$\frac{2\sigma_w^2 L_w}{\pi V} \cdot \frac{1 + \frac{8}{3} \left( 2.678 L_w \frac{\omega}{V} \right)^2}{\left[ 1 + \left( 2.678 L_w \frac{\omega}{V} \right)^2 \right]^{11/6}}$
$\Phi_q(\omega)$	$\frac{\pm \left( \frac{\omega}{V} \right)^2}{1 + \left( \frac{4b\omega}{\pi V} \right)^2} \cdot \Phi_w(\omega)$	$\frac{\pm \left( \frac{\omega}{V} \right)^2}{1 + \left( \frac{4b\omega}{\pi V} \right)^2} \cdot \Phi_w(\omega)$

The variable  $b$  represents the aircraft wingspan. The variables  $L_u$ ,  $L_v$ ,  $L_w$  represent the turbulence scale lengths. The variables  $\sigma_u$ ,  $\sigma_v$ ,  $\sigma_w$  represent the turbulence intensities:

# Von Karman Wind Turbulence Model (Continuous)

---

The spectral density definitions of turbulence angular rates are defined in the references as three variations, which are displayed in the following table:

$$p_g = \frac{\partial w_g}{\partial y} \quad q_g = \frac{\partial w_g}{\partial x} \quad r_g = -\frac{\partial v_g}{\partial x}$$

$$p_g = \frac{\partial w_g}{\partial y} \quad q_g = \frac{\partial w_g}{\partial x} \quad r_g = \frac{\partial v_g}{\partial x}$$

$$p_g = -\frac{\partial w_g}{\partial y} \quad q_g = -\frac{\partial w_g}{\partial x} \quad r_g = \frac{\partial v_g}{\partial x}$$

The variations affect only the vertical ( $q_g$ ) and lateral ( $r_g$ ) turbulence angular rates.

Keep in mind that the longitudinal turbulence angular rate spectrum,  $\Phi_p(\omega)$ , is a rational function. The rational function is derived from curve-fitting a complex algebraic function, not the vertical turbulence velocity spectrum,  $\Phi_w(\omega)$ , multiplied by a scale factor. Because the turbulence angular rate spectra contribute less to the aircraft gust response than the turbulence velocity spectra, it may explain the variations in their definitions.

The variations lead to the following combinations of vertical and lateral turbulence angular rate spectra.

<b>Vertical</b>	<b>Lateral</b>
$\Phi_q(\omega)$	$-\Phi_r(\omega)$
$\Phi_q(\omega)$	$\Phi_r(\omega)$
$-\Phi_q(\omega)$	$\Phi_r(\omega)$

# Von Karman Wind Turbulence Model (Continuous)

To generate a signal with the correct characteristics, a unit variance, band-limited white noise signal is passed through forming filters. The forming filters are approximations of the Von Kármán velocity spectra which are valid in a range of normalized frequencies of less than 50 radians. These filters can be found in both the Military Handbook MIL-HDBK-1797 and the reference by Ly and Chan.

The following two tables display the transfer functions.

<b>MIL-F-8785C</b>	
<b>Longitudinal</b>	
$H_u(s)$	$\frac{\sigma_u \sqrt{\frac{2}{\pi}} \cdot \frac{L_u}{V} \left( 1 + 0.25 \frac{L_u}{V} s \right)}{1 + 1.357 \frac{L_u}{V} s + 0.1987 \left( \frac{L_u}{V} \right)^2 s^2}$
$H_p(s)$	$\sigma_w \sqrt{\frac{0.8}{V}} \cdot \frac{\left( \frac{\pi}{4b} \right)^{1/6}}{L_w^{1/3} \left( 1 + \left( \frac{4b}{\pi V} \right) s \right)}$
<b>Lateral</b>	
$H_v(s)$	$\frac{\sigma_v \sqrt{\frac{1}{\pi}} \cdot \frac{L_v}{V} \left( 1 + 2.7478 \frac{L_v}{V} s + 0.3398 \left( \frac{L_v}{V} \right)^2 s^2 \right)}{1 + 2.9958 \frac{L_v}{V} s + 1.9754 \left( \frac{L_v}{V} \right)^2 s^2 + 0.1539 \left( \frac{L_v}{V} \right)^3 s^3}$

# Von Karman Wind Turbulence Model (Continuous)

<b>MIL-F-8785C</b>	
$H_r(s)$	$\frac{\mp \frac{s}{V}}{\left(1 + \left(\frac{3b}{\pi V}\right)s\right)} \cdot H_v(s)$
<b>Vertical</b>	
$H_w(s)$	$\frac{\sigma_w \sqrt{\frac{1}{\pi}} \cdot \frac{L_w}{V} \left(1 + 2.7478 \frac{L_w}{V} s + 0.3398 \left(\frac{L_w}{V}\right)^2 s^2\right)}{1 + 2.9958 \frac{L_w}{V} s + 1.9754 \left(\frac{L_w}{V}\right)^2 s^2 + 0.1539 \left(\frac{L_w}{V}\right)^3 s^3}$
$H_q(s)$	$\frac{\pm \frac{s}{V}}{\left(1 + \left(\frac{4b}{\pi V}\right)s\right)} \cdot H_w(s)$
<b>MIL-HDBK-1797</b>	
<b>Longitudinal</b>	
$H_u(s)$	$\frac{\sigma_u \sqrt{\frac{2}{\pi}} \cdot \frac{L_u}{V} \left(1 + 0.25 \frac{L_u}{V} s\right)}{1 + 1.357 \frac{L_u}{V} s + 0.1987 \left(\frac{L_u}{V}\right)^2 s^2}$

# Von Karman Wind Turbulence Model (Continuous)

MIL-HDBK-1797	
$H_p(s)$	$\sigma_w \sqrt{\frac{0.8}{V}} \cdot \frac{\left(\frac{\pi}{4b}\right)^{1/6}}{(2L_w)^{1/3} \left(1 + \left(\frac{4b}{\pi V}\right)s\right)}$
<b>Lateral</b>	
$H_v(s)$	$\frac{\sigma_v \sqrt{\frac{1}{\pi} \cdot \frac{2L_v}{V}} \left(1 + 2.7478 \frac{2L_v}{V} s + 0.3398 \left(\frac{2L_v}{V}\right)^2 s^2\right)}{1 + 2.9958 \frac{2L_v}{V} s + 1.9754 \left(\frac{2L_v}{V}\right)^2 s^2 + 0.1539 \left(\frac{2L_v}{V}\right)^3 s^3}$
$H_r(s)$	$\frac{\mp \frac{s}{V}}{\left(1 + \left(\frac{3b}{\pi V}\right)s\right)} \cdot H_v(s)$
<b>Vertical</b>	

# Von Karman Wind Turbulence Model (Continuous)

	MIL-HDBK-1797
$H_w(s)$	$\frac{\sigma_w \sqrt{\frac{1}{\pi}} \cdot \frac{2L_w}{V} \left( 1 + 2.7478 \frac{2L_w}{V} s + 0.3398 \left( \frac{2L_w}{V} \right)^2 s^2 \right)}{1 + 2.9958 \frac{2L_w}{V} s + 1.9754 \left( \frac{2L_w}{V} \right)^2 s^2 + 0.1539 \left( \frac{2L_w}{V} \right)^3 s^3}$
$H_q(s)$	$\frac{\pm \frac{s}{V}}{\left( 1 + \left( \frac{4b}{\pi V} \right) s \right)} \cdot H_w(s)$

Divided into two distinct regions, the turbulence scale lengths and intensities are functions of altitude.

**Note** The same transfer functions result after evaluating the turbulence scale lengths. The differences in turbulence scale lengths and turbulence transfer functions balance offset.

## Low-Altitude Model (Altitude < 1000 feet)

According to the military references, the turbulence scale lengths at low altitudes, where  $h$  is the altitude in feet, are represented in the following table:

MIL-F-8785C	MIL-HDBK-1797
$L_w = h$ $L_u = L_v = \frac{h}{(0.177 + 0.000823h)^{1.2}}$	$2L_w = h$ $L_u = 2L_v = \frac{h}{(0.177 + 0.000823h)^{1.2}}$

# Von Karman Wind Turbulence Model (Continuous)

The turbulence intensities are given below, where  $W_{20}$  is the wind speed at 20 feet (6 m). Typically for light turbulence, the wind speed at 20 feet is 15 knots; for moderate turbulence, the wind speed is 30 knots; and for severe turbulence, the wind speed is 45 knots.

$$\sigma_w = 0.1W_{20}$$
$$\frac{\sigma_u}{\sigma_w} = \frac{\sigma_v}{\sigma_w} = \frac{1}{(0.177 + 0.000823h)^{0.4}}$$

The turbulence axes orientation in this region is defined as follows:

- Longitudinal turbulence velocity,  $u_g$ , aligned along the horizontal relative mean wind vector
- Vertical turbulence velocity,  $w_g$ , aligned with vertical.

At this altitude range, the output of the block is transformed into body coordinates.

## Medium/High Altitudes (Altitude > 2000 feet)

For medium to high altitudes the turbulence scale lengths and intensities are based on the assumption that the turbulence is isotropic. In the military references, the scale lengths are represented by the following equations:

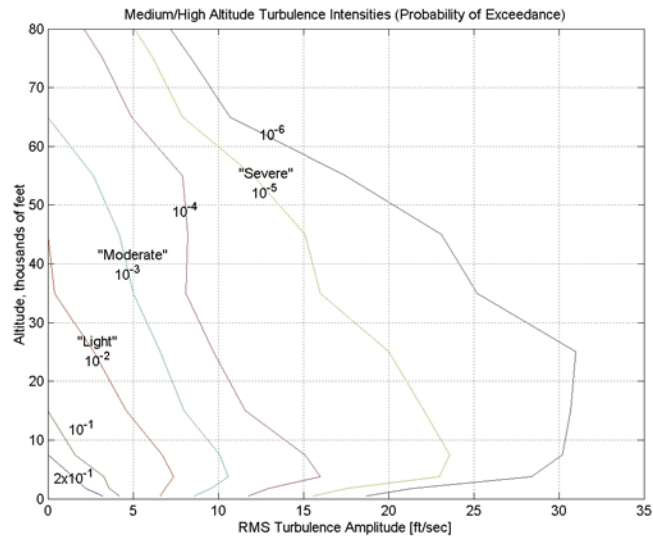
MIL-F-8785C	MIL-HDBK-1797
$L_u = L_v = L_w = 2500 \text{ ft}$	$L_u = 2L_v = 2L_w = 2500 \text{ ft}$

The turbulence intensities are determined from a lookup table that provides the turbulence intensity as a function of altitude and the probability of the turbulence intensity being exceeded. The relationship of the turbulence intensities is represented in the following equation:

$$\sigma_u = \sigma_v = \sigma_w$$

The turbulence axes orientation in this region is defined as being aligned with the body coordinates:

# Von Karman Wind Turbulence Model (Continuous)



## Between Low and Medium/High Altitudes (1000 feet < Altitude < 2000 feet)

At altitudes between 1000 feet and 2000 feet, the turbulence velocities and turbulence angular rates are determined by linearly interpolating between the value from the low altitude model at 1000 feet transformed from mean horizontal wind coordinates to body coordinates and the value from the high altitude model at 2000 feet in body coordinates.



# Von Karman Wind Turbulence Model (Continuous)

## Dialog Box

Function Block Parameters: Von Karman Wind Turbulence Model (Continuous)

Wind Turbulence Model (mask) (link)

Generate atmospheric turbulence. White noise is passed through a filter to give the turbulence the specified velocity spectra.

Medium/high altitude scale lengths from the specifications are 762 m (2500 ft) for Von Karman turbulence and 533.4 m (1750 ft) for Dryden turbulence.

Parameters

Units: Metric (MKS)

Specification: MIL-F-8785C

Model type: Continuous Von Karman (+q -r)

Wind speed at 6 m defines the low-altitude intensity (m/s):  
15

Wind direction at 6 m (degrees clockwise from north):  
0

Probability of exceedance of high-altitude intensity: 10<sup>-2</sup> - Light

Scale length at medium/high altitudes (m):  
762

Wingspan (m):  
10

Band limited noise sample time (sec):  
0.1

Noise seeds [ug vg wg pg]:  
[23341 23342 23343 23344]

Turbulence on

OK Cancel Help Apply

## Units

Define the units of wind speed due to the turbulence.

# Von Karman Wind Turbulence Model (Continuous)

---

<b>Units</b>	<b>Wind Velocity</b>	<b>Altitude Air Speed</b>	
Metric (MKS)	Meters/second	Meters	Meters/second
English (Velocity in ft/s)	Feet/second	Feet	Feet/second
English (Velocity in kts)	Knots	Feet	Knots

## Specification

Define which military reference to use. This affects the application of turbulence scale lengths in the lateral and vertical directions

## Model type

Select the wind turbulence model to use:

Continuous Von Karman (+q -r)	Use continuous representation of Von Kármán velocity spectra with positive vertical and negative lateral angular rates spectra.
Continuous Von Karman (+q +r)	Use continuous representation of Von Kármán velocity spectra with positive vertical and lateral angular rates spectra.
Continuous Von Karman (-q +r)	Use continuous representation of Von Kármán velocity spectra with negative vertical and positive lateral angular rates spectra.

# Von Karman Wind Turbulence Model (Continuous)

---

Continuous Dryden (+q -r)	Use continuous representation of Dryden velocity spectra with positive vertical and negative lateral angular rates spectra.
Continuous Dryden (+q +r)	Use continuous representation of Dryden velocity spectra with positive vertical and lateral angular rates spectra.
Continuous Dryden (-q +r)	Use continuous representation of Dryden velocity spectra with negative vertical and positive lateral angular rates spectra.
Discrete Dryden (+q -r)	Use discrete representation of Dryden velocity spectra with positive vertical and negative lateral angular rates spectra.
Discrete Dryden (+q +r)	Use discrete representation of Dryden velocity spectra with positive vertical and lateral angular rates spectra.
Discrete Dryden (-q +r)	Use discrete representation of Dryden velocity spectra with negative vertical and positive lateral angular rates spectra.

The Continuous Von Kármán selections conform to the transfer function descriptions.

## **Wind speed at 6 m defines the low altitude intensity**

The measured wind speed at a height of 20 feet (6 meters) provides the intensity for the low-altitude turbulence model.

# Von Karman Wind Turbulence Model (Continuous)

---

## **Wind direction at 6 m (degrees clockwise from north)**

The measured wind direction at a height of 20 feet (6 meters) is an angle to aid in transforming the low-altitude turbulence model into a body coordinates.

## **Probability of exceedance of high-altitude intensity**

Above 2000 feet, the turbulence intensity is determined from a lookup table that gives the turbulence intensity as a function of altitude and the probability of the turbulence intensity's being exceeded.

## **Scale length at medium/high altitudes**

The turbulence scale length above 2000 feet is assumed constant, and from the military references, a figure of 1750 feet is recommended for the longitudinal turbulence scale length of the Dryden spectra.

---

**Note** An alternate scale length value changes the power spectral density asymptote and gust load.

---

## **Wingspan**

The wingspan is required in the calculation of the turbulence on the angular rates.

## **Band-limited noise sample time (seconds)**

The sample time at which the unit variance white noise signal is generated.

## **Noise seeds**

There are four random numbers required to generate the turbulence signals, one for each of the three velocity components and one for the roll rate. The turbulences on the pitch and yaw angular rates are based on further shaping of the outputs from the shaping filters for the vertical and lateral velocities.

## **Turbulence on**

Selecting the check box generates the turbulence signals.

# Von Karman Wind Turbulence Model (Continuous)

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the altitude in units selected.
Second		Contains the aircraft speed in units selected.
Third		Contains a direction cosine matrix.

Output	Dimension Type	Description
First	Three-element signal	Contains the turbulence velocities, in the selected units.
Second	Three-element signal	Contains the turbulence angular rates, in radians per second.

## Assumptions and Limitations

The frozen turbulence field assumption is valid for the cases of mean-wind velocity and the root-mean-square turbulence velocity, or intensity, are small relative to the aircraft's ground speed.

The turbulence model describes an average of all conditions for clear air turbulence because the following factors are not incorporated into the model:

- Terrain roughness
- Lapse rate
- Wind shears
- Mean wind magnitude
- Other meteorological factors (except altitude)

## References

- U.S. Military Handbook MIL-HDBK-1797, 19 December 1997.  
U.S. Military Specification MIL-F-8785C, 5 November 1980.

# Von Karman Wind Turbulence Model (Continuous)

---

Chalk, C., Neal, P., Harris, T., Pritchard, F., Woodcock, R., "Background Information and User Guide for MIL-F-8785B(ASG), 'Military Specification-Flying Qualities of Piloted Airplanes'," AD869856, Cornell Aeronautical Laboratory, August 1969.

Hoblit, F., *Gust Loads on Aircraft: Concepts and Applications*, AIAA Education Series, 1988.

Ly, U., Chan, Y., "Time-Domain Computation of Aircraft Gust Covariance Matrices," AIAA Paper 80-1615, Atmospheric Flight Mechanics Conference, Danvers, MA., August 11-13, 1980.

McRuer, D., Ashkenas, I., Graham, D., *Aircraft Dynamics and Automatic Control*, Princeton University Press, July 1990.

Moorhouse, D., Woodcock, R., "Background Information and User Guide for MIL-F-8785C, 'Military Specification-Flying Qualities of Piloted Airplanes'," ADA119421, Flight Dynamic Laboratory, July 1982.

McFarland, R., "A Standard Kinematic Model for Flight Simulation at NASA-Ames," NASA CR-2497, Computer Sciences Corporation, January 1975.

Tatom, F., Smith, R., Fichtl, G., "Simulation of Atmospheric Turbulent Gusts and Gust Gradients," AIAA Paper 81-0300, Aerospace Sciences Meeting, St. Louis, MO., January 12-15, 1981.

Yeager, J., "Implementation and Testing of Turbulence Models for the F18-HARV Simulation," NASA CR-1998-206937, Lockheed Martin Engineering & Sciences, March 1998.

## See Also

Dryden Wind Turbulence Model (Continuous)

Dryden Wind Turbulence Model (Discrete)

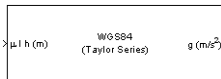
Discrete Wind Gust Model

Wind Shear Model

**Purpose** Implement 1984 World Geodetic System (WGS84) representation of Earth's gravity

**Library** Environment/Gravity

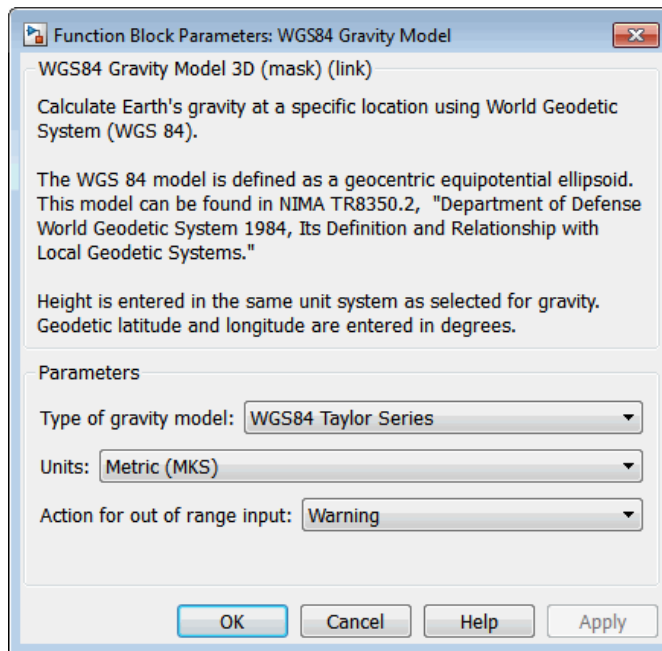
## Description



The WGS84 Gravity Model block implements the mathematical representation of the geocentric equipotential ellipsoid of the World Geodetic System (WGS84). The block output is the Earth's gravity at a specific location. Gravity precision is controlled via the **Type of gravity model** parameter.

The WGS84 Gravity Model block icon displays the input and output units selected from the **Units** list.

## Dialog Box



# WGS84 Gravity Model

---

## Type of gravity model

Specifies the method to calculate gravity:

- WGS84 Taylor Series
- WGS84 Close Approximation
- WGS84 Exact

## Units

Specifies the input and output units:

<b>Units</b>	<b>Height</b>	<b>Gravity</b>
Metric (MKS)	Meters	Meters per second squared
English	Feet	Feet per second squared

## Exclude Earth's atmosphere

Select for the value for the Earth's gravitational field to exclude the mass of the atmosphere.

Clear for the value for the Earth's gravitational field to include the mass of the atmosphere.

This option is available only with **Type of gravity model WGS84 Close Approximation** or **WGS84 Exact**.

## Precessing reference frame

When selected, the angular velocity of the Earth is calculated using the International Astronomical Union (IAU) value of the Earth's angular velocity and the precession rate in right ascension. To obtain the precession rate in right ascension, Julian centuries from Epoch J2000.0 is calculated using the dialog parameters of **Month**, **Day**, and **Year**.

If cleared, the angular velocity of the Earth used is the value of the standard Earth rotating at a constant angular velocity.



This option is available only with **Type of gravity model WGS84 Close Approximation** or **WGS84 Exact**.

## **Input Julian date**

When selected, another input port, JD, appears on the block mask. Select this check box if you want to manually specify the Julian date for the block. Otherwise, the block calculates the Julian date given the values of **Month**, **Day**, and **Year**. Selecting this block disables the **Month**, **Day**, and **Year** parameters.

## **Month**

Specifies the month used to calculate Julian centuries from Epoch J2000.0.

This option is available only with **Type of gravity model WGS84 Close Approximation** or **WGS84 Exact** and only when **Preprocessing reference frame** is selected. It is disabled if you select **Input Julian Date**.

## **Day**

Specifies the day used to calculate Julian centuries from Epoch J2000.0.

This option is available only with **Type of gravity model WGS84 Close Approximation** or **WGS84 Exact** and only when **Preprocessing reference frame** is selected. It is disabled if you select **Input Julian Date**.

## **Year**

Specifies the year used to calculate Julian centuries from Epoch J2000.0. The year must be 2000 or greater.

This option is available only with **Type of gravity model WGS84 Close Approximation** or **WGS84 Exact** and only when **Preprocessing reference frame** is selected. It is disabled if you select **Input Julian Date**.

# WGS84 Gravity Model

---

## No centrifugal effects

When selected, calculated gravity is based on pure attraction resulting from the normal gravitational potential.

If cleared, calculated gravity includes the centrifugal force resulting from the Earth's angular velocity.

This option is available only with **Type of gravity model** **WGS84 Close Approximation** or **WGS84 Exact**.

## Action for out of range input

Specify if out-of-range input invokes a warning, error, or no action.

## Inputs and Outputs

Input	Dimension Type	Description
First	Three-element vector	Contains the position in geodetic latitude, longitude and altitude, with units in degrees, degrees, and selected units of length respectively.
Second (Optional)	Scalar	Contains the Julian centuries that you specified.

Output	Dimension Type	Description
Output	Vector	Applies to gravity in the north-east-down (NED) coordinate system. The Exact method should output both normal and tangent gravity (down and north in the NED coordinate system).

## **Assumptions and Limitations**

The WGS84 gravity calculations are based on the assumption of a geocentric equipotential ellipsoid of revolution. Since the gravity potential is assumed to be the same everywhere on the ellipsoid, there must be a specific theoretical gravity potential that can be uniquely determined from the four independent constants defining the ellipsoid.

Use of the WGS84 Taylor Series model should be limited to low geodetic heights. It is sufficient near the surface when submicrogal precision is not necessary. At medium and high geodetic heights, it is less accurate.

Use of the WGS84 Close Approximation model should be limited to a geodetic height of 20,000.0 m (approximately 65,620.0 feet). Below this height, it gives results with submicrogal precision.

To predict and determine a satellite orbit with high accuracy, use the EGM96 through degree and order 70.

## **Examples**

See the Environment Models subsystem in the Airframe subsystem of the aeroblk\_HL20 model for an example of this block.

## **Reference**

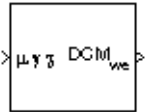
[1] NIMA TR8350.2: "Department of Defense World Geodetic System 1984, Its Definition and Relationship with Local Geodetic Systems."

# Wind Angles to Direction Cosine Matrix

**Purpose** Convert wind angles to direction cosine matrix

**Library** Utilities/Axes Transformations

**Description**



The Wind Angles to Direction Cosine Matrix block converts three wind rotation angles into a 3-by-3 direction cosine matrix (DCM). The DCM matrix performs the coordinate transformation of a vector in earth axes ( $ox_0, oy_0, oz_0$ ) into a vector in wind axes ( $ox_3, oy_3, oz_3$ ). The order of the axis rotations required to bring this about is:

- 1 A rotation about  $oz_0$  through the heading angle ( $\chi$ ) to axes ( $ox_1, oy_1, oz_1$ )
- 2 A rotation about  $oy_1$  through the flight path angle ( $\gamma$ ) to axes ( $ox_2, oy_2, oz_2$ )
- 3 A rotation about  $ox_2$  through the bank angle ( $\mu$ ) to axes ( $ox_3, oy_3, oz_3$ )

$$\begin{bmatrix} ox_3 \\ oy_3 \\ oz_3 \end{bmatrix} = DCM_{we} \begin{bmatrix} ox_0 \\ oy_0 \\ oz_0 \end{bmatrix}$$

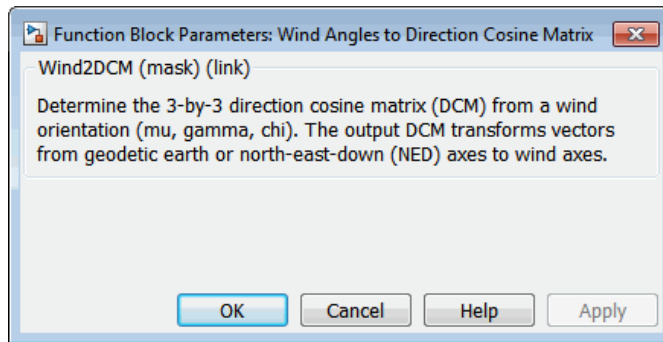
$$\begin{bmatrix} ox_3 \\ oy_3 \\ oz_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \mu & \sin \mu \\ 0 & -\sin \mu & \cos \mu \end{bmatrix} \begin{bmatrix} \cos \gamma & 0 & -\sin \gamma \\ 0 & 1 & 0 \\ \sin \gamma & 0 & \cos \gamma \end{bmatrix} \begin{bmatrix} \cos \chi & \sin \chi & 0 \\ -\sin \chi & \cos \chi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} ox_0 \\ oy_0 \\ oz_0 \end{bmatrix}$$

Combining the three axis transformation matrices defines the following DCM.

$$DCM_{we} = \begin{bmatrix} \cos \gamma \cos \chi & \cos \gamma \sin \chi & -\sin \gamma \\ (\sin \mu \sin \gamma \cos \chi - \cos \mu \sin \chi) & (\sin \mu \sin \gamma \sin \chi + \cos \mu \cos \chi) & \sin \mu \cos \gamma \\ (\cos \mu \sin \gamma \cos \chi + \sin \mu \sin \chi) & (\cos \mu \sin \gamma \sin \chi - \sin \mu \cos \chi) & \cos \mu \cos \gamma \end{bmatrix}$$

# Wind Angles to Direction Cosine Matrix

## Dialog Box



## Inputs and Outputs

Input	Dimension Type	Description
First	3-by-1 vector	Contains wind angles, in radians.

Output	Dimension Type	Description
First	3-by-3 direction cosine matrix	Transforms earth vectors to wind vectors.

## Assumptions and Limitations

This implementation generates a flight path angle that lies between  $\pm 90$  degrees, and bank and heading angles that lie between  $\pm 180$  degrees.

## See Also

Direction Cosine Matrix Body to Wind  
Direction Cosine Matrix to Rotation Angles  
Direction Cosine Matrix to Wind Angles  
Rotation Angles to Direction Cosine Matrix

# Wind Angular Rates

## Purpose

Calculate wind angular rates from body angular rates, angle of attack, sideslip angle, rate of change of angle of attack, and rate of change of sideslip

## Library

Flight Parameters

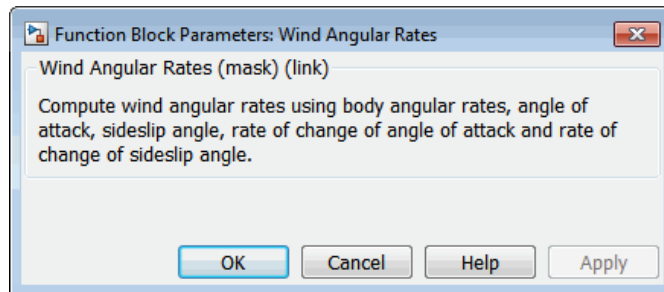
## Description



The Wind Angular Rates block supports the equations of motion in wind-fixed frame models by calculating the wind-fixed angular rates ( $p_w, q_w, r_w$ ). The body-fixed angular rates ( $p_b, q_b, r_b$ ), angle of attack ( $\alpha$ ), sideslip angle ( $\beta$ ), rate of change of angle of attack ( $\dot{\alpha}$ ), and rate of change of sideslip ( $\dot{\beta}$ ) are related to the wind-fixed angular rate by the following equation.

$$\begin{bmatrix} p_w \\ q_w \\ r_w \end{bmatrix} = \begin{bmatrix} \cos \alpha \cos \beta & \sin \beta & \sin \alpha \cos \beta \\ -\cos \alpha \sin \beta & \cos \beta & -\sin \alpha \sin \beta \\ -\sin \alpha & 0 & \cos \alpha \end{bmatrix} \begin{bmatrix} p_b - \dot{\beta} \sin \alpha \\ q_b - \dot{\alpha} \\ r_b + \dot{\beta} \cos \alpha \end{bmatrix}$$

## Dialog Box



## Inputs and Outputs

Input	Dimension Type	Description
First	2-by-1 vector	Contains angle of attack and sideslip, in radians.
	2-by-1 vector	Contains rate of change of angle of attack and rate of change of sideslip, in radians per second.
		Contains the body angular rates, in radians per second.

Output	Dimension Type	Description
First		Contains the wind angular rates, in radians per second.

## See Also

3DoF (Body Axes)

6DoF Wind (Quaternion)

6DoF Wind (Wind Angles)

Custom Variable Mass 3DoF (Body Axes)

Custom Variable Mass 6DoF Wind (Quaternion)

Custom Variable Mass 6DoF Wind (Wind Angles)

Simple Variable Mass 3DoF (Body Axes)

Simple Variable Mass 6DoF Wind (Quaternion)

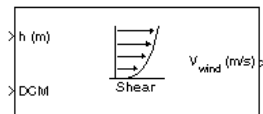
Simple Variable Mass 6DoF Wind (Wind Angles)

# Wind Shear Model

**Purpose** Calculate wind shear conditions

**Library** Environment/Wind

## Description



The Wind Shear Model block adds wind shear to the aerospace model. This implementation is based on the mathematical representation in the Military Specification MIL-F-8785C [1]. The magnitude of the wind shear is given by the following equation for the mean wind profile as a function of altitude and the measured wind speed at 20 feet (6 m) above the ground.

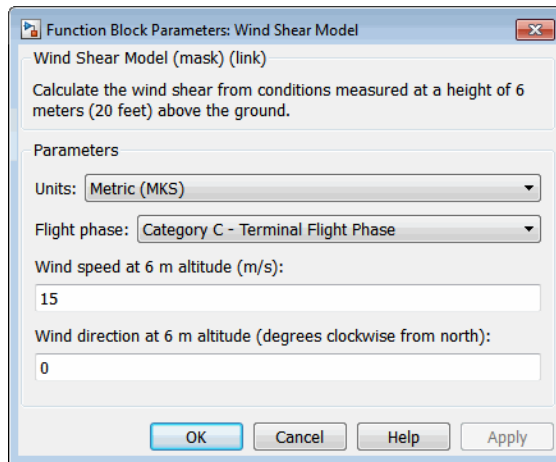
$$u_w = W_{20} \frac{\ln\left(\frac{h}{z_0}\right)}{\ln\left(\frac{20}{z_0}\right)}, \quad 3ft < h < 1000ft$$

where  $u_w$  is the mean wind speed,  $W_{20}$  is the measured wind speed at an altitude of 20 feet,  $h$  is the altitude, and  $z_0$  is a constant equal to 0.15 feet for Category C flight phases and 2.0 feet for all other flight phases. Category C flight phases are defined in reference [1] to be terminal flight phases, which include takeoff, approach, and landing.

The resultant mean wind speed in the flat Earth axis frame is changed to body-fixed axis coordinates by multiplying by the direction cosine matrix (DCM) input to the block. The block output is the mean wind speed in the body-fixed axis.



## Dialog Box



### Units

Define the units of wind shear.

Units	Wind	Altitude
Metric (MKS)	Meters/second	Meters
English (Velocity in ft/s)	Feet/second	Feet
English (Velocity in kts)	Knots	Feet

### Flight phase

Select flight phase:

- Category C — Terminal Flight Phases
- Other

### Wind speed at 6 m (20 feet) altitude (m/s, f/s, or knots)

The measured wind speed at an altitude of 20 feet (6 m) above the ground.

# Wind Shear Model

---

## Wind direction at 6 m (20 feet) altitude (degrees clockwise from north)

The direction of the wind at an altitude of 20 feet (6 m), measured in degrees clockwise from the direction of the Earth  $x$ -axis (north). The wind direction is defined as the direction from which the wind is coming.

### Inputs and Outputs

Input	Dimension Type	Description
First		Contains the altitude in units selected.
Second	3-by-3 direction cosine matrix	

Output	Dimension Type	Description
First	3-by-1 vector	Contains the mean wind speed in the body axes frame, in the selected units.

### Examples

See Wind Shear Model in aeroblk\_HL20 for an example of this block.

### Reference

U.S. Military Specification MIL-F-8785C, 5 November 1980.

### See Also

Discrete Wind Gust Model

Dryden Wind Turbulence Model (Continuous)

Dryden Wind Turbulence Model (Discrete)

Von Karman Wind Turbulence Model (Continuous)

## Purpose

Calculate Earth's magnetic field at specific location and time using World Magnetic Model 2000 (WMM2000)

---

**Note** World Magnetic Model 2000 will be removed in a future version. For model years between 2000 and the start of 2010, use International Geomagnetic Reference Field 11. For model years between 2010 and the start of 2015, use World Magnetic Model 2010.

---

## Library

Environment/Gravity

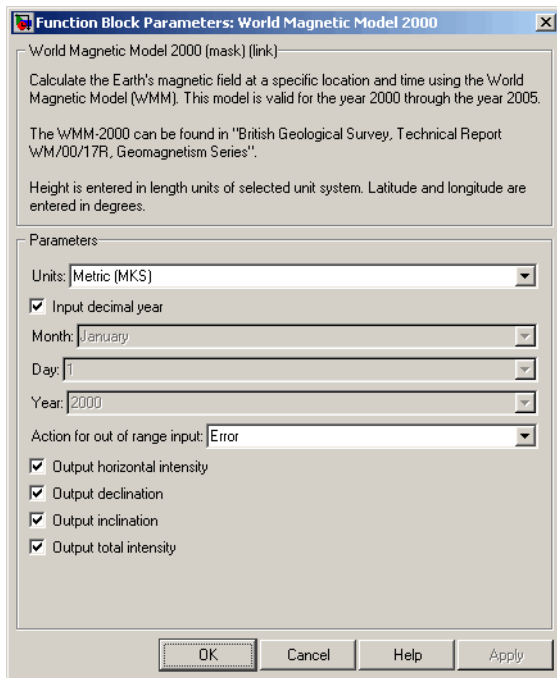
## Description

h (m)	Magnetic Field (nT)
$\mu$ (deg)	Horizontal Intensity (nT)
$\delta$ (deg)	Declination (deg)
$I$ (deg)	Inclination (deg)
Decimal Year	Total Intensity (nT)

The WMM2000 block implements the mathematical representation of the National Geospatial Intelligence Agency (NGA) World Magnetic Model 2000. The WMM2000 block calculates the Earth's magnetic field vector, horizontal intensity, declination, inclination, and total intensity at a specified location and time. The reference frame is north-east-down (NED).

# World Magnetic Model 2000

## Dialog Box



### Units

Specifies the input and output units:

Units	Height	Magnetic Field	Horizontal Intensity	Total Intensity
Metric (MKS)	Meters	Nanotesla	Nanotesla	Nanotesla
English	Feet	Nanogauss	Nanogauss	Nanogauss

### Input decimal year

When selected, the decimal year is an input for the World Magnetic Model 2000 block. Otherwise, a date must be specified using the dialog parameters of **Month**, **Day**, and **Year**.

**Month**

Specifies the month used to calculate decimal year.

**Day**

Specifies the day used to calculate decimal year.

**Year**

Specifies the year used to calculate decimal year.

**Action for out of range input**

Specify if out-of-range input invokes a warning, error or no action.

**Output horizontal intensity**

When selected, the horizontal intensity is output.

**Output declination**

When selected, the declination, the angle between true north and the magnetic field vector (positive eastwards), is output.

**Output inclination**

When selected, the inclination, the angle between the horizontal plane and the magnetic field vector (positive downwards), is output.

**Output total intensity**

When selected, the total intensity is output.

## Inputs and Outputs

Input	Dimension Type	Description
First		Contains the height, in selected units.
Second		Contains the latitude in degrees.

# World Magnetic Model 2000

---

Input	Dimension Type	Description
Third		Contains the longitude in degrees.
Fourth (Optional)		Contains the desired year in a decimal format to include any fraction of the year that has already passed. The value is the current year plus the number of days that have passed in this year divided by 365. The following code illustrates how to calculate the decimal year, dyear, for March 21, 2005:  %%BEGIN CODE%% dyear=decyear('21-March-2005','dd-mmm-yyyy') %%END CODE%%

Output	Dimension Type	Description
First		Contains the magnetic field vector in selected units.
Second (Optional)		Contains the horizontal intensity in selected units.
Third (Optional)		Contains the declination in degrees.
Fourth (Optional)		Contains the inclination in degrees.
Fifth (Optional)		Contains the total intensity in selected units.

## Limitations

The WMM2000 specification produces data that is reliable five years after the epoch of the model, which is January 1, 2000.

The internal calculation of decimal year does not take into account local time or leap seconds.

The WMM2000 specification describes only the long-wavelength spatial magnetic fluctuations due to the Earth's core. Intermediate and short-wavelength fluctuations, contributed from the crustal field (the mantle and crust), are not included. Also, the substantial fluctuations of the geomagnetic field, which occur constantly during magnetic storms and almost constantly in the disturbance field (auroral zones), are not included.

## Reference

Macmillan, S. and J. M. Quinn, 2000. "The Derivation of the World Magnetic Model 2000," *British Geological Survey Technical Report WM/00/17R*.

<http://www.ngdc.noaa.gov/geomag/WMM/DoDWMM.shtml>

## See Also

International Geomagnetic Reference Field 11, World Magnetic Model 2010

# World Magnetic Model 2005

---

## Purpose

Calculate Earth's magnetic field at specific location and time using World Magnetic Model 2005 (WMM2005)

---

**Note** World Magnetic Model 2005 will be removed in a future version. For model years between 2000 and the start of 2010, use International Geomagnetic Reference Field 11. For model years between 2010 and the start of 2015, use World Magnetic Model 2010.

---

## Library

Environment/Gravity

## Description

> h (m)	Magnetic Field (nT)
> μ (deg)	Horizontal Intensity (nT)
> I (deg)	Declination (deg)
> I (deg)	Inclination (deg)
> Decimal Year	Total Intensity (nT)

The WMM2005 block implements the mathematical representation of the National Geospatial Intelligence Agency (NGA) World Magnetic Model 2005. The WMM2005 block calculates the Earth's magnetic field vector, horizontal intensity, declination, inclination, and total intensity at a specified location and time. The reference frame is north-east-down (NED).

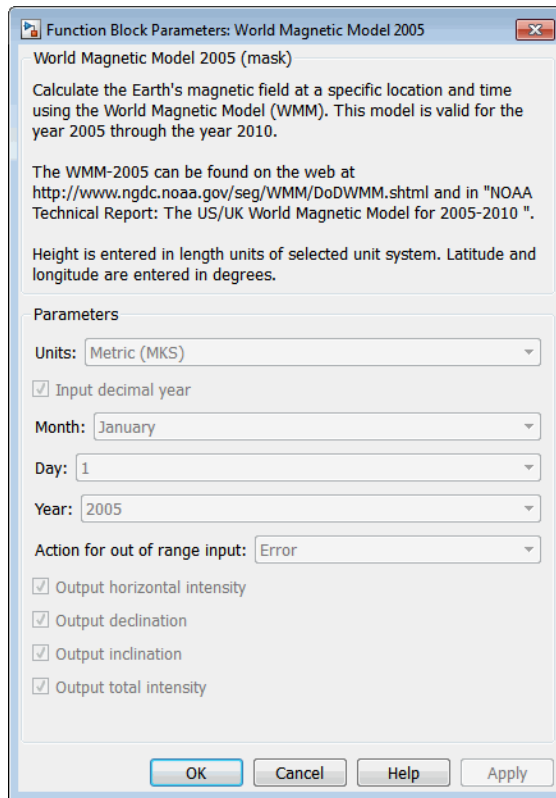
---

**Note** You cannot use this block to model the Earth magnetic field above an altitude of 1,000,000 meters.

---



## Dialog Box



### Units

Specifies the input and output units:

Units	Height	Magnetic Field	Horizontal Intensity	Total Intensity
Metric (MKS)	Meters	Nanotesla	Nanotesla	Nanotesla
English	Feet	Nanogauss	Nanogauss	Nanogauss

# World Magnetic Model 2005

---

## **Input decimal year**

When selected, the decimal year is an input for the World Magnetic Model 2005 block. Otherwise, a date must be specified using the dialog parameters of **Month**, **Day**, and **Year**.

## **Month**

Specifies the month used to calculate decimal year.

## **Day**

Specifies the day used to calculate decimal year.

## **Year**

Specifies the year used to calculate decimal year.

## **Action for out of range input**

Specify if out-of-range input invokes a warning, error or no action.

## **Output horizontal intensity**

When selected, the horizontal intensity is output.

## **Output declination**

When selected, the declination, the angle between true north and the magnetic field vector (positive eastwards), is output.

## **Output inclination**

When selected, the inclination, the angle between the horizontal plane and the magnetic field vector (positive downwards), is output.

## **Output total intensity**

When selected, the total intensity is output.

## **Inputs and Outputs**

<b>Input</b>	<b>Dimension Type</b>	<b>Description</b>
First		Contains the height, in selected units.
Second		Contains the latitude in degrees.

Input	Dimension Type	Description
Third		Contains the longitude in degrees.
Fourth (Optional)		Contains the desired year in a decimal format to include any fraction of the year that has already passed. The value is the current year plus the number of days that have passed in this year divided by 365. The following code illustrates how to calculate the decimal year, dyear, for March 21, 2005:  %%BEGIN CODE%% dyear=decyear('21-March-2005','dd-mmm-yyyy') %%END CODE%%

Output	Dimension Type	Description
First	Vector	Contains the magnetic field in selected units.
Second (Optional)		Contains the horizontal intensity in selected units.
Third (Optional)		Contains the declination in degrees.
Fourth (Optional)		Contains the inclination in degrees.
Fifth (Optional)		Contains the total intensity in selected units.

## Limitations

The WMM2005 specification produces data that is reliable five years after the epoch of the model, which is January 1, 2005.

# World Magnetic Model 2005

---

The internal calculation of decimal year does not take into account local time or leap seconds.

The WMM2005 specification describes only the long-wavelength spatial magnetic fluctuations due to the Earth's core. Intermediate and short-wavelength fluctuations, contributed from the crustal field (the mantle and crust), are not included. Also, the substantial fluctuations of the geomagnetic field, which occur constantly during magnetic storms and almost constantly in the disturbance field (auroral zones), are not included.

## **Reference**

<http://www.ngdc.noaa.gov/geomag/WMM/DoDWMM.shtml>

## **See Also**

International Geomagnetic Reference Field 11, World Magnetic Model 2010

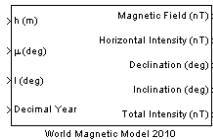
## Purpose

Calculate Earth's magnetic field at specific location and time using World Magnetic Model 2010 (WMM2010)

## Library

Environment/Gravity

## Description



The WMM2010 block implements the mathematical representation of the National Geospatial Intelligence Agency (NGA) World Magnetic Model 2010. The WMM2010 block calculates the Earth's magnetic field vector, horizontal intensity, declination, inclination, and total intensity at a specified location and time. The reference frame is north-east-down (NED).

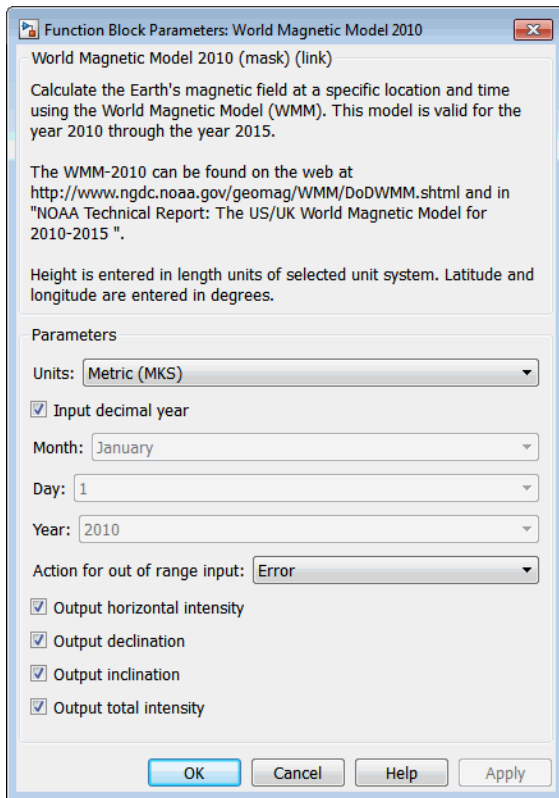
---

**Note** You cannot use this block to model the Earth magnetic field above an altitude of 1,000,000 meters.

---

# World Magnetic Model 2010

## Dialog Box



## Units

Specifies the input and output units:

<b>Units</b>	<b>Height</b>	<b>Magnetic Field</b>	<b>Horizontal Intensity</b>	<b>Total Intensity</b>
Metric (MKS)	Meters	Nanotesla	Nanotesla	Nanotesla
English	Feet	Nanogauss	Nanogauss	Nanogauss

## **Input decimal year**

When selected, the decimal year is an input for the World Magnetic Model 2010 block. Otherwise, a date must be specified using the dialog parameters of **Month**, **Day**, and **Year**.

## **Month**

Specifies the month used to calculate decimal year.

## **Day**

Specifies the day used to calculate decimal year.

## **Year**

Specifies the year used to calculate decimal year.

## **Action for out of range input**

Specify if out-of-range input invokes a warning, error or no action.

## **Output horizontal intensity**

When selected, the horizontal intensity is output.

## **Output declination**

When selected, the declination, the angle between true north and the magnetic field vector (positive eastwards), is output.

## **Output inclination**

When selected, the inclination, the angle between the horizontal plane and the magnetic field vector (positive downwards), is output.

## **Output total intensity**

When selected, the total intensity is output.

## **Inputs and Outputs**

<b>Input</b>	<b>Dimension Type</b>	<b>Description</b>
First		Contains the height, in selected units.
Second		Contains the latitude in degrees.

# World Magnetic Model 2010

---

Input	Dimension Type	Description
Third		Contains the longitude in degrees.
Fourth (Optional)		Contains the desired year in a decimal format to include any fraction of the year that has already passed. The value is the current year plus the number of days that have passed in this year divided by 365.  The following code illustrates how to calculate the decimal year, dyear, for March 21, 2010:  %%BEGIN CODE%% dyear=decyear('21-March-2010','dd-mmm-yyyy') %%END CODE%%

Output	Dimension Type	Description
First	Vector	Contains the magnetic field in selected units.
Second (Optional)		Contains the horizontal intensity in selected units.
Third (Optional)		Contains the declination in degrees.
Fourth (Optional)		Contains the inclination in degrees.
Fifth (Optional)		Contains the total intensity in selected units.

## Limitations

The WMM2010 specification produces data that is reliable five years after the epoch of the model, which is January 1, 2015.



The internal calculation of decimal year does not take into account local time or leap seconds.

The WMM2010 specification describes only the long-wavelength spatial magnetic fluctuations due to the Earth's core. Intermediate and short-wavelength fluctuations, contributed from the crustal field (the mantle and crust), are not included. Also, the substantial fluctuations of the geomagnetic field, which occur constantly during magnetic storms and almost constantly in the disturbance field (auroral zones), are not included.

**Reference**

<http://www.ngdc.noaa.gov/geomag/WMM/DoDWMM.shtml>

**See Also**

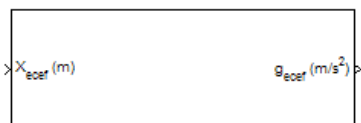
World Magnetic Model 2000, World Magnetic Model 2005

# Zonal Harmonic Gravity Model

**Purpose** Calculate zonal harmonic representation of planetary gravity

**Library** Environment/Gravity

## Description



The Zonal Harmonic Gravity Model block calculates the zonal harmonic representation of planetary gravity at a specific location based on planetary gravitational potential. This block provides a convenient way to describe the gravitational field of a planet outside its surface.

By default, the block uses the fourth order zonal coefficient for Earth to calculate the zonal harmonic gravity. It also allows you to specify the second or third zonal coefficient.

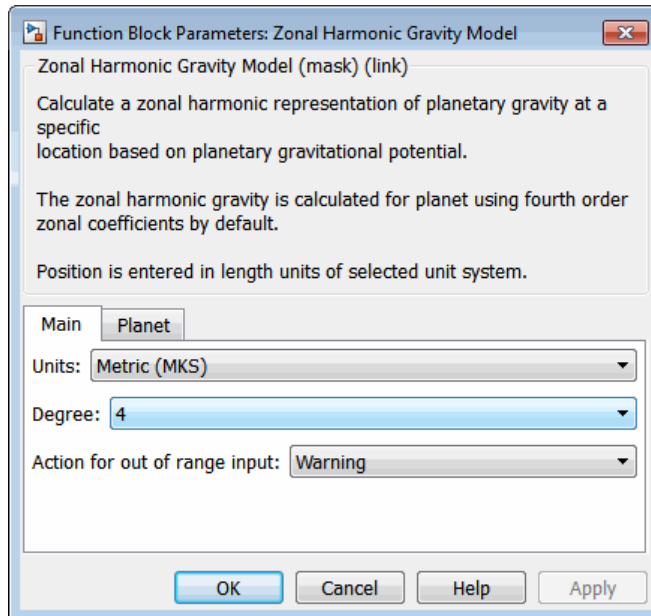
gravityzonal is implemented using the following planetary parameter values for each planet:

Planet	Equatorial Radius ( $R_e$ ) in Meters	Gravitational Parameter ( $GM$ ) in $m^3/s^2$	Zonal Harmonic Coefficients (J Values)
Earth	6378.1363e3	3.986004415e14	[ 0.0010826269 -0.0000025323 -0.0000016204 ]
Jupiter	71492.e3	1.268e17	[0.01475 0 -0.00058]
Mars	3397.2e3	4.305e13	[ 0.001964 0.000036 ]
Mercury	2439.0e3	2.2032e13	0.00006
Moon	1738.0e3	4902.799e9	0.0002027
Neptune	24764e3	6.809e15	0.004
Saturn	60268.e3	3.794e16	[0.01645 0 -0.001]

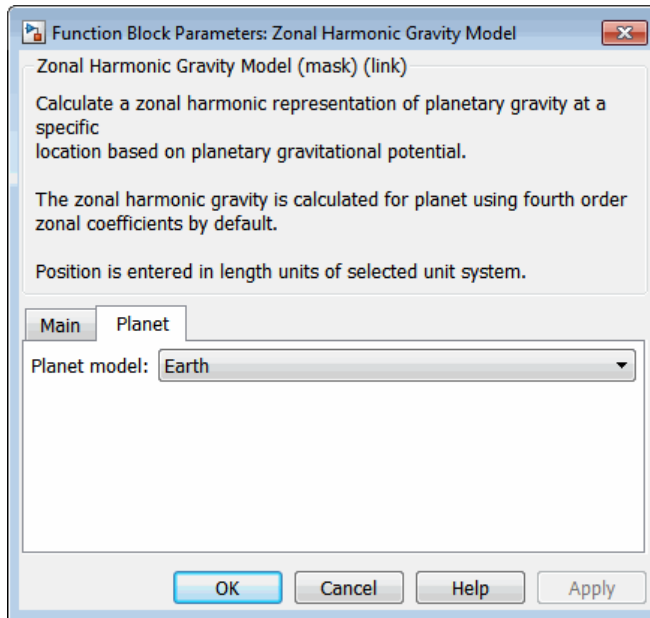
# Zonal Harmonic Gravity Model

Planet	Equatorial Radius (Re) in Meters	Gravitational Parameter (GM) in $m^3/s^2$	Zonal Harmonic Coefficients (J Values)
Uranus	25559.e3	5.794e15	0.012
Venus	6052.0e3	3.257e14	0.000027

## Dialog Box



# Zonal Harmonic Gravity Model



## Units

Specify the input units:

<b>Units</b>	<b>Position</b>	<b>Equatorial Radius</b>	<b>Gravitational Parameter</b>
Metric (MKS)	Meters	Meters	Meters cubed per second squared
English	Feet	Feet	Feet cubed per second squared

## Degree

Specify the degree of harmonic model.

- 2 — Second degree, J2. Most significant or largest spherical harmonic term, which accounts for the oblateness of a planet.

# Zonal Harmonic Gravity Model

---

- 3 — Third degree, J3.
- 4 — Fourth degree, J4 (default).

## Action for out of range input

Specify if out-of-range input invokes a warning, error, or no action.

## Planet model

Specify the planetary model. From the list, select Mercury, Venus, Earth, Moon, Mars, Jupiter, Saturn, Uranus, Neptune, or Custom.

Selecting Custom enables you to specify your own planetary model. This option enables the **Equatorial radius**, **Gravitational parameter**, and **J values** parameters.

Selecting Mercury, Venus, Moon, Uranus, or Neptune limits the degree to 2.

Selecting Mars limits the degree to 3.

## Equatorial radius

Specify the planetary equatorial radius in the length units that the **Units** parameter defines.

## Gravitational parameter

Specify the planetary gravitational parameter in the length units cubed per second squared that the **Units** parameter defines.

## J values

Specify a 3-element array that defines the zonal harmonic coefficients.

# Zonal Harmonic Gravity Model

---

## Inputs and Outputs

This block accepts only scalar inputs ( $m=1$ ).

Input	Dimension Type	Description
First	m-by-3 matrix	Contains planet-centered planet-fixed coordinates from the center of the planet in the selected length units. If <b>Planet model</b> has a value of <b>Earth</b> , this matrix contains Earth-centered Earth-fixed (ECEF) coordinates.

Output	Dimension Type	Description
First	m-by-3 array	Contains gravity values in the $x$ -axis, $y$ -axis and $z$ -axis of the planet-centered planet-fixed coordinates in the selected length units per second squared.

## References

Vallado, D. A., *Fundamentals of Astrodynamics and Applications*, McGraw-Hill, New York, 1997.

Fortescue, P., J. Stark, G. Swinerd, (Eds.). *Spacecraft Systems Engineering*, Third Edition, Wiley & Sons, West Sussex, 2003.

Tewari, A., *Atmospheric and Space Flight Dynamics Modeling and Simulation with MATLAB and Simulink*, Birkhäuser, Boston, 2007.

# Aerospace Units

---

The main blocks of the Aerospace Blockset library support standard measurement systems. The Unit Conversion blocks support all units listed in this table.

Quantity	Metric (MKS)	English
Acceleration	meters/second <sup>2</sup> (m/s <sup>2</sup> ), kilometers/second <sup>2</sup> (km/s <sup>2</sup> ), (kilometers/hour)/second (km/h-s), g-unit ( <i>g</i> )	inches/second <sup>2</sup> (in/s <sup>2</sup> ), feet/second <sup>2</sup> (ft/s <sup>2</sup> ), (miles/hour)/second (mph/s), g-unit ( <i>g</i> )
Angle	radian (rad), degree (deg), revolution	radian (rad), degree (deg), revolution
Angular acceleration	radians/second <sup>2</sup> (rad/s <sup>2</sup> ), degrees/second <sup>2</sup> (deg/s <sup>2</sup> ), revolutions/minute (rpm), revolutions/second (rps)	radians/second <sup>2</sup> (rad/s <sup>2</sup> ), degrees/second <sup>2</sup> (deg/s <sup>2</sup> ), revolutions/minute (rpm), revolutions/second (rps)
Angular velocity	radians/second (rad/s), degrees/second (deg/s), revolutions/minute (rpm)	radians/second (rad/s), degrees/second (deg/s), revolutions/minute (rpm)
Density	kilogram/meter <sup>3</sup> (kg/m <sup>3</sup> )	pound mass/foot <sup>3</sup> (lbm/ft <sup>3</sup> ), slug/foot <sup>3</sup> (slug/ft <sup>3</sup> ), pound mass/inch <sup>3</sup> (lbm/in <sup>3</sup> )
Force	newton (N)	pound (lb)
Inertia	kilogram-meter <sup>2</sup> (kg-m <sup>2</sup> )	slug-foot <sup>2</sup> (slug-ft <sup>2</sup> )
Length	meter (m)	inch (in), foot (ft), mile (mi), nautical mile (nm)

<b>Quantity</b>	<b>Metric (MKS)</b>	<b>English</b>
Mass	kilogram (kg)	slug (slug), pound mass (lbm)
Pressure	Pascal (Pa)	pound/inch <sup>2</sup> (psi), pound/foot <sup>2</sup> (psf), atmosphere (atm)
Temperature	kelvin (K), degrees Celsius (°C)	degrees Fahrenheit (°F), degrees Rankine (°R)
Torque	newton-meter (N-m)	pound-feet (lb-ft)
Velocity	meters/second (m/s), kilometers/second (km/s), kilometers/hour (km/h)	inches/second (in/s), feet/second (ft/s), feet/minute (ft/min), miles/hour (mph), knots



## Symbols and Numerics

- 1D Controller  $[A(v), B(v), C(v), D(v)]$  block 4-2
- 1D Controller Blend  $u=(1-L).K1.y+L.K2.y$   
block[oneD Controller Blend  
 $u=(1-L).K1.y+L.K2.y$  block 4-6
- 1D Observer Form  $[A(v), B(v), C(v), F(v), H(v)]$   
block 4-10
- 1D Self-Conditioned  $[A(v), B(v), C(v), D(v)]$   
block 4-14
- 2D Controller  $[A(v), B(v), C(v), D(v)]$  block 4-18
- 2D Controller Blend block 4-22
- 2D Observer Form  $[A(v), B(v), C(v), F(v), H(v)]$   
block 4-26
- 2D Self-Conditioned  $[A(v), B(v), C(v), D(v)]$   
block 4-31
- 3D Controller  $[A(v), B(v), C(v), D(v)]$  block 4-36
- 3D Observer Form  $[A(v), B(v), C(v), F(v), H(v)]$   
block 4-40
- 3D Self-Conditioned  $[A(v), B(v), C(v), D(v)]$   
block 4-45
- 3DoF (Body Axes) block 4-54
- 3DoF (Wind Axes) block 4-59
- 3DoF Animation block 4-50
- 3x3 Cross Product block 4-65
- 4th Order Point Mass (Longitudinal) block 4-67
- 4th Order Point Mass Forces (Longitudinal)  
block 4-71
- 6DoF (Euler Angles) block 4-78
- 6DoF (Quaternion) block 4-85
- 6DoF Animation block 4-74
- 6DoF ECEF (Quaternion) block 4-91
- 6DoF Wind (Quaternion) block 4-102
- 6DoF Wind (Wind Angles) block 4-109
- 6th Order Point Mass (Coordinated Flight)  
block 4-116
- 6th Order Point Mass Forces (Coordinated  
Flight) block 4-121

## A

- AC3D coordinates 2-20
- Acceleration Conversion block 4-124
- Adjoint of 3x3 Matrix block 4-126
- Aerodynamic Forces and Moments block 4-128
- airspeed correction 3-2
- Angle Conversion block 4-133
- Angular Acceleration Conversion block 4-135
- Angular Velocity Conversion block 4-137

## B

- Besselian Epoch to Julian Epoch block 4-139
- body coordinates 2-14

## C

- Calculate Range block 4-141
- CIRA Atmosphere Model block 4-144
- COESA Atmosphere Model block 4-149
- Controllers
  - 1D Controller  $[A(v), B(v), C(v), D(v)]$   
block[oneD Controller  
 $[A(v), B(v), C(v), D(v)]$  4-2
- coordinate systems
  - display 2-19
  - modeling 2-14
  - navigation 2-16
  - overview 2-12
- Create 3x3 Matrix block 4-153
- creating an aerospace model
  - basic steps 2-2
- Crossover Pilot Model 4-155
- Custom Variable Mass 3DoF (Body Axes)  
block 4-162
- Custom Variable Mass 3DoF (Wind Axes)  
block 4-167
- Custom Variable Mass 6DoF (Euler Angles)  
block 4-173

- Custom Variable Mass 6DoF (Quaternion)  
block 4-181
- Custom Variable Mass 6DoF ECEF (Quaternion)  
block 4-187
- Custom Variable Mass 6DoF Wind (Quaternion)  
block 4-198
- Custom Variable Mass 6DoF Wind (Wind Angles)  
block 4-205

## D

- Density Conversion block 4-211
- Determinant of 3x3 Matrix block 4-213
- Digital DATCOM Forces and Moments  
block 4-215
- Direction Cosine Matrix Body to Wind  
block 4-222
- Direction Cosine Matrix Body to Wind to Alpha  
and Beta block 4-224
- Direction Cosine Matrix ECEF to NED  
block 4-227
- Direction Cosine Matrix ECEF to NED to  
Latitude and Longitude block 4-230
- Direction Cosine Matrix to Quaternions  
block 4-233
- Direction Cosine Matrix to Rotation Angles  
block 4-235
- Direction Cosine Matrix to Wind Angles  
block 4-238
- Discrete Wind Gust Model block 4-241
- Dryden Wind Turbulence Model (Continuous)  
block 4-246
- Dryden Wind Turbulence Model (Discrete)  
block 4-260
- Dynamic Pressure block 4-273

## E

- Earth Nutation 4-274
- ECEF coordinates 2-18

- ECEF Position to LLA block 4-278
- ECI coordinates 2-17
- Estimate Center of Gravity block 4-286
- Estimate Inertia Tensor block 4-288

## F

- Flat Earth to LLA block 4-290
- FlightGear
  - aircraft models 2-29
  - example 2-42
  - flight simulator overview 2-22
  - installing 2-26
  - obtaining 2-22
  - running 2-34
- FlightGear coordinates 2-19
- FlightGear Preconfigured 6DoF Animation  
block 4-295
- Force Conversion block 4-298

## G

- Gain Scheduled Lead-Lag block 4-300
- Generate Run Script block 4-302
- Geocentric to Geodetic Latitude block 4-309
- Geodetic to Geocentric Latitude block 4-316

## H

- Horizontal Wind Model block 4-323

## I

- Ideal Airspeed Correction block 4-326
- Incidence & Airspeed block 4-330
- Incidence, Sideslip & Airspeed block 4-332
- Interpolate Matrix(x) block 4-340
- Interpolate Matrix(x,y) block 4-343
- Interpolate Matrix(x,y,z) block 4-346
- Invert 3x3 Matrix block 4-350
- ISA Atmosphere Model block 4-351

**J**

Julian Epoch to Besselian Epoch block 4-353

**L**

Lapse Rate Model block 4-356  
latitude 2-16  
Length Conversion block 4-361  
lifting body (HL-20) 3-20  
LLA to ECEF Position block 4-365  
LLA to Flat Earth block 4-369

**M**

Mach Number block 4-375  
Mass Conversion block 4-377  
MATLAB  
    opening examples  
        using the command line 1-16  
MATLAB files  
    running simulations from 2-11  
Moments about CG due to Forces block 4-383  
Moon Libration 4-385

**N**

NED coordinates 2-17  
Non-Standard Day 210C block 4-389  
Non-Standard Day 310 block 4-396  
NRLMSISE-00 Model block 4-406

**P**

Pack net\_fdm Packet for FlightGear block 4-414  
parameters  
    tuning 2-10  
Pilot Joystick All block 4-438  
Pilot Joystick block 4-434  
Planetary Ephemeris 4-442  
Precision Pilot Model 4-448  
Pressure Altitude block 4-453

Pressure Conversion block 4-455

**Q**

Quaternion Conjugate block 4-457  
Quaternion Division block 4-459  
Quaternion Inverse block 4-461  
Quaternion Modulus block 4-463  
Quaternion Multiplication block 4-465  
Quaternion Norm block 4-467  
Quaternion Normalize block 4-469  
Quaternion Rotation block 4-471  
Quaternions to Direction Cosine Matrix  
    block 4-473  
Quaternions to Rotation Angles block 4-475

**R**

Radius at Geocentric Latitude block 4-478  
Receive net\_ctrl Packet from FlightGear  
    block 4-482  
Relative Ratio block 4-487  
Rotation Angles to Direction Cosine Matrix  
    block 4-490

**S**

Second Order Linear Actuator block 4-363  
Second Order Nonlinear Actuator block 4-403  
Self-Conditioned [A,B,C,D] block 4-495  
Send net\_fdm Packet to FlightGear block 4-500  
Simple Variable Mass 3DoF (Body Axes)  
    block 4-503  
Simple Variable Mass 3DoF (Wind Axes)  
    block 4-510  
Simple Variable Mass 6DoF (Euler Angles)  
    block 4-517  
Simple Variable Mass 6DoF (Quaternion)  
    block 4-526  
Simple Variable Mass 6DoF ECEF (Quaternion)  
    block 4-533

- Simple Variable Mass 6DoF Wind (Quaternion)
  - block 4-545
- Simple Variable Mass 6DoF Wind (Wind Angles)
  - block 4-554
- Simulation Pace block 4-562
- simulations
  - running from MATLAB file 2-11
- Simulink
  - modifying models 1-13
  - opening examples
    - using the Help browser 1-16
    - using the Run button 1-16
  - running examples 1-11
  - using the Simulink Library Browser 2-2
- SinCos block 4-565
- Symmetric Inertia Tensor block 4-573

## **T**

- Temperature Conversion block 4-575
- Three-Axis Accelerometer block 4-577
- Three-axis Gyroscope block 4-583
- Three-Axis Inertial Measurement Unit
  - block 4-588
- tuning parameters 2-10
- Turbofan Engine System block 4-596

- Tustin Pilot Model 4-600

## **U**

- Unpack net\_ctrl Packet from FlightGear
  - block 4-603

## **V**

- Velocity Conversion block 4-624
- Von Kármán Wind Turbulence Model (Continuous) block 4-626

## **W**

- WGS84 Gravity Model block 4-641
- Wind Angles to Direction Cosine Matrix
  - block 4-646
- Wind Angular Rates block 4-648
- wind coordinates 2-15
- Wind Shear Model block 4-650
- World Magnetic Model 2000 block 4-653
- World Magnetic Model 2005 block 4-658
- World Magnetic Model 2010 block 4-663
- Wright Flyer 3-10