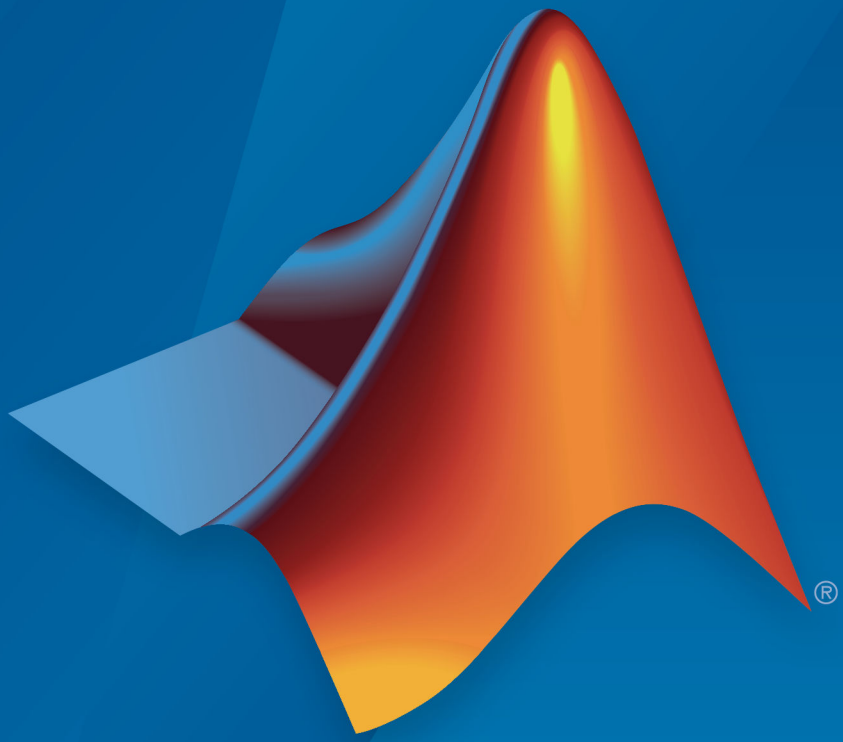
Aerospace Blockset™

User's Guide



MATLAB® & SIMULINK®

R2018a

MathWorks®

How to Contact MathWorks

Latest news: `www.mathworks.com`

Sales and services: `www.mathworks.com/sales_and_services`

User community: `www.mathworks.com/matlabcentral`

Technical support: `www.mathworks.com/support/contact_us`

Phone: 508-647-7000

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

Aerospace Blockset ™ User's Guide

© COPYRIGHT 2002–2018 by The MathWorks, Inc.

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

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

Trademarks

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

Patents

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

Revision History

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)
September 2013	Online only	Revised for Version 3.12 (Release 2013b)
March 2014	Online only	Revised for Version 3.13 (Release 2014a)
October 2014	Online only	Revised for Version 3.14 (Release 2014b)
March 2015	Online only	Revised for Version 3.15 (Release 2015a)
September 2015	Online only	Revised for Version 3.16 (Release 2015b)
October 2015	Online only	Rereleased for Version 3.15.1 (Release 2015aSP1)
March 2016	Online only	Revised for Version 3.17 (Release 2016a)
September 2016	Online only	Revised for Version 3.18 (Release 2016b)
March 2017	Online only	Revised for Version 3.19 (Release 2017a)
September 2017	Online only	Revised for Version 3.20 (Release 2017b)
March 2018	Online only	Revised for Version 3.21 (Release 2018a)

Contents

1 Getting Started

Aerospace Blockset Product Description 1-2
Key Features 1-2

Code Generation Support 1-3

Support for Aerospace Toolbox Quaternion Functions 1-4

Related Products 1-5
Requirements for the Aerospace Blockset Product 1-5
Other Related Products 1-5

Explore the NASA HL-20 Model 1-6
Introduction 1-6
What This Example Illustrates 1-6
Open the Model 1-7
Key Subsystems 1-9
NASA HL-20 Example 1-10
Modify the Model 1-13

Open Aerospace Examples 1-15

2 Aerospace Blockset Software

Create Aerospace Models 2-2
Basic Steps 2-2

Build a Simple Actuator System 2-4
Build the Model 2-4

Run the Simulation 2-8

About Aerospace Coordinate Systems 2-10
Fundamental Coordinate System Concepts 2-10
Coordinate Systems for Modeling 2-11
Coordinate Systems for Navigation 2-14
Coordinate Systems for Display 2-16
References 2-18

Flight Simulator Interface 2-19
About the FlightGear Interface 2-19
Supported FlightGear Versions 2-19
Obtain FlightGear 2-20
Configure Your Computer for FlightGear 2-20
FlightGear and Video Cards in Windows Systems 2-21
Install and Start FlightGear 2-21
Install Additional FlightGear Scenery 2-23

Work with the Flight Simulator Interface 2-24
Introduction 2-24
About Aircraft Geometry Models 2-25
Work with Aircraft Geometry Models 2-27
Run FlightGear with the Simulink Models 2-29
Run the HL-20 Example with FlightGear 2-36
Send and Receive Data 2-38

Simulink Projects Template for Flight Simulation Applications 2-43
Flight Simulation Applications 2-43

Flight Instrument Gauges 2-46

Display Measurements with Cockpit Instruments 2-47

Calculate UT1 to UTC Values 2-50
Use the Delta UT1 Block to Create Difference Values for the Direction Cosine Matrix ECI to ECEF Block 2-50

3 Case Studies

Ideal Airspeed Correction **3-2**
Introduction **3-2**
Airspeed Correction Models **3-2**
Measure Airspeed **3-3**
Model Airspeed Correction **3-4**
Simulate Airspeed Correction **3-7**

1903 Wright Flyer **3-9**
Introduction **3-9**
Wright Flyer Model **3-10**
Airframe Subsystem **3-10**
Environment Subsystem **3-13**
Pilot Subsystem **3-14**
Run the Simulation **3-15**
References **3-16**

NASA HL-20 Lifting Body Airframe **3-18**
Introduction **3-18**
NASA HL-20 Lifting Body **3-18**
The HL-20 Airframe and Controller Model **3-19**
References **3-28**

4 Blocks — Alphabetical List

A Aerospace Units

Getting Started

- “Aerospace Blockset 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 1-15

Aerospace Blockset 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?” (Simulink)
- `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`; 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 1-9 “NASA HL-20 Example” on page 1-10 “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 3-18.

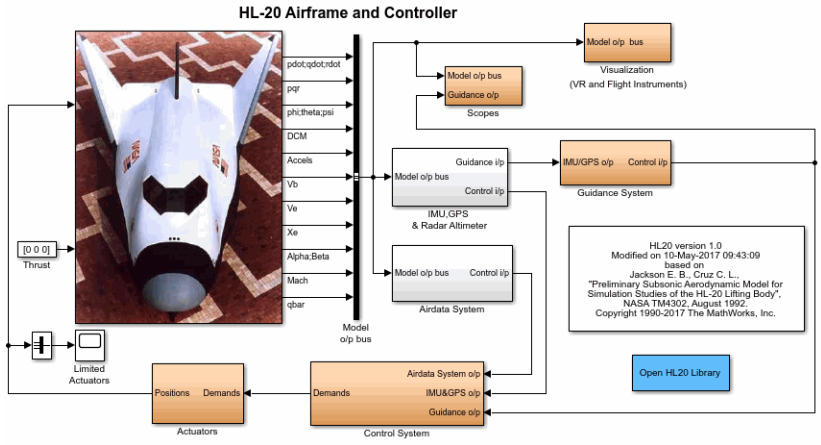
What This Example Illustrates

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

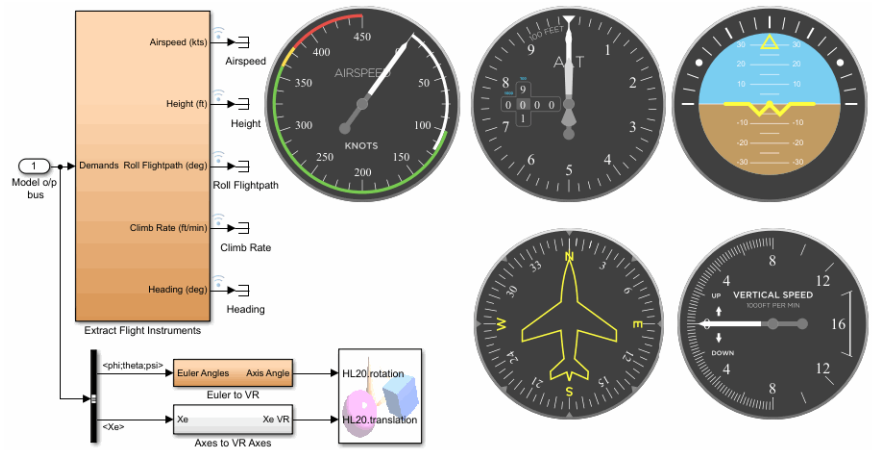
- Representing bodies and their degrees of freedom with the Equations of Motion library blocks
- Using the Aerospace Blockset blocks with other Simulink blocks
- Feeding Simulink signals to and from Aerospace Blockset blocks with Actuator and Sensor blocks
- Encapsulating groups of blocks into subsystems
- Visualizing an aircraft with Simulink 3D Animation™ and Aerospace Blockset Flight Instrument library blocks.

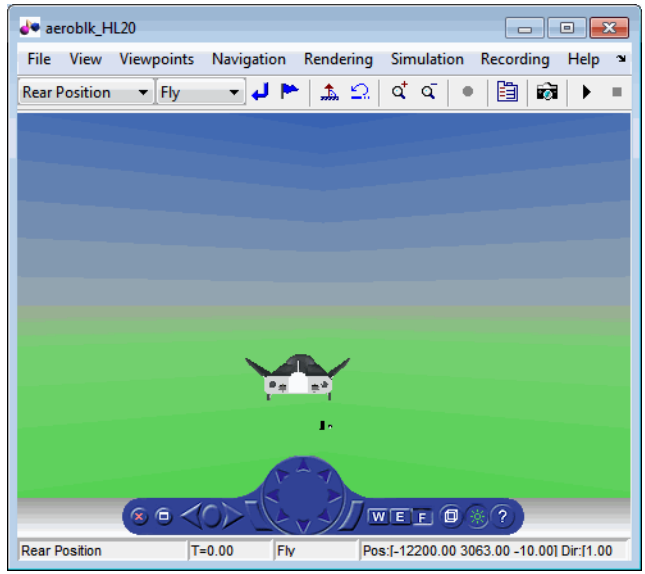
Open the Model

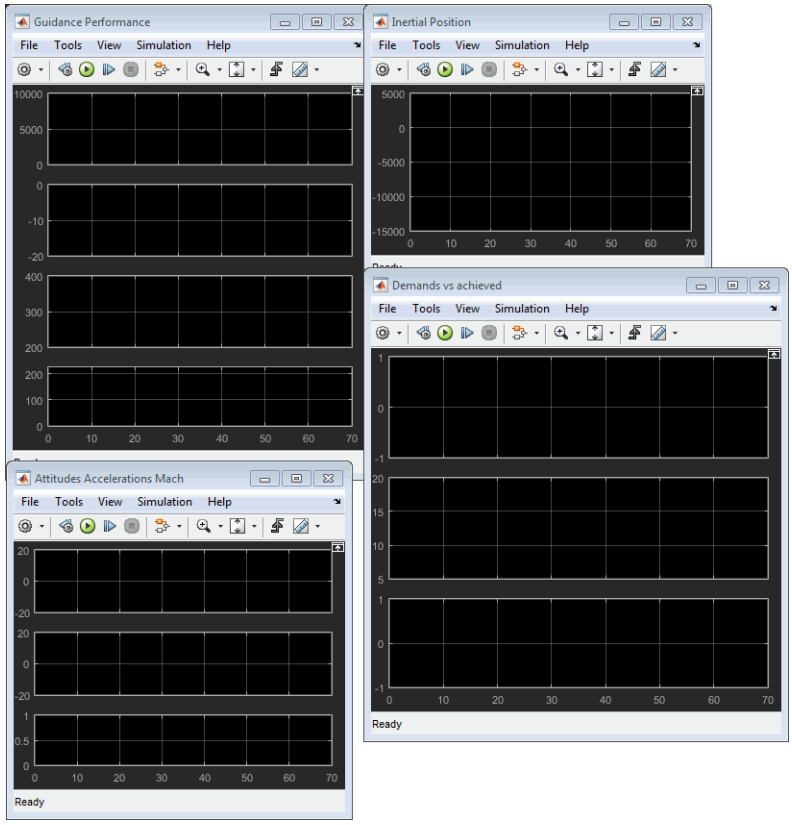
To open the NASA HL-20 airframe example, type the example name, `aeroblk_HL20`, at the MATLAB command line. The model opens.



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




aeroblk_HL20
File View Viewpoints Navigation Rendering Simulation Recording Help
Rear Position
Fly
W E F
Rear Position
T=0.00
Fly
Pos:[-12200.00 3063.00 -10.00] Dir:[1.00



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.

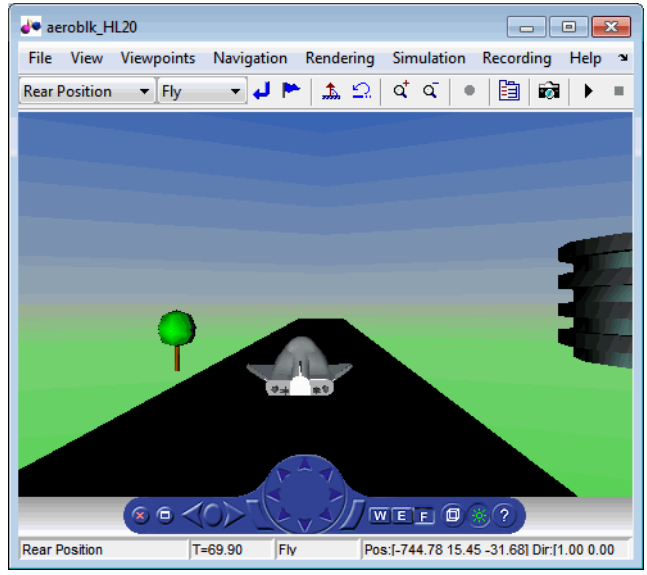
- The Forces and Moments subsystem implements the Aerodynamic Forces and Moments block. This subsystem calculates body forces and body moments.
- The Aerodynamic Coefficients subsystem implements several subsystems to calculate six aerodynamic coefficients.

NASA HL-20 Example

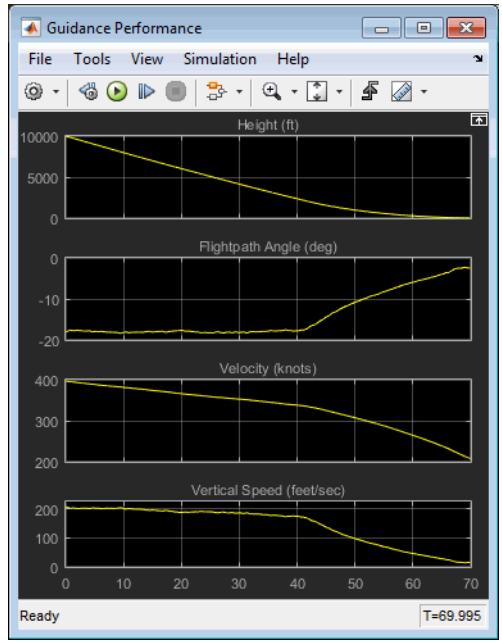
Running an example lets you observe the model simulation in real time. After you run the example, you can examine the resulting data in plots, graphs, and other visualization tools. To run this model, follow these steps:

1. If it is not already open, open the `aeroblk_HL20` example.
2. From the **Simulation** menu, select **Start**. On Microsoft® Windows® systems, you can also click the **Start** button in the model window toolbar.

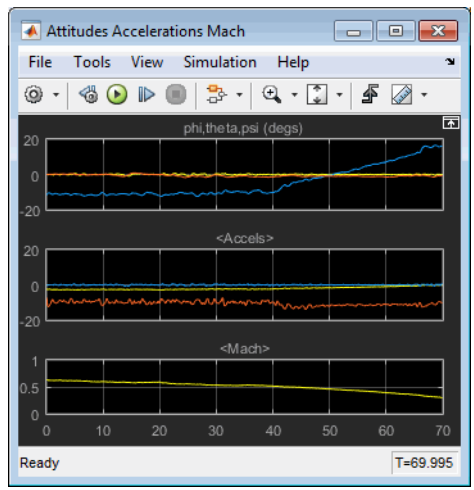
   The simulation proceeds until the aircraft lands:



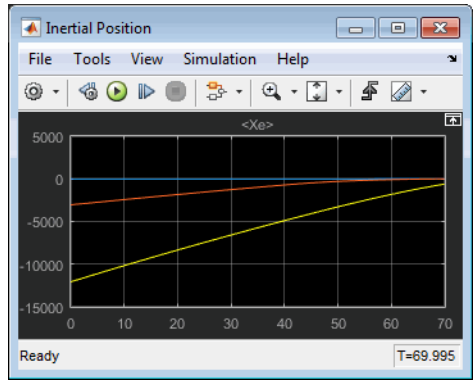
View of the landed airframe



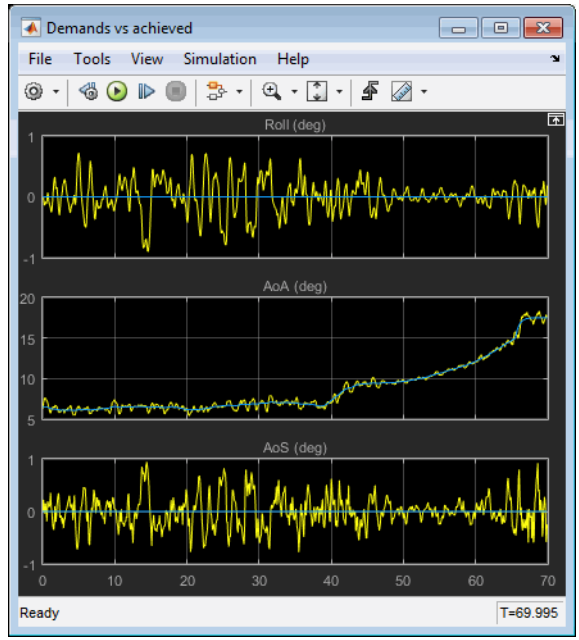
Plot that Measures Guidance Performance



Plot that Measures Altitude Accelerations Mach



Plot that Measures Inertial Position



Plot that Measures Demand Data Against Achieved Data

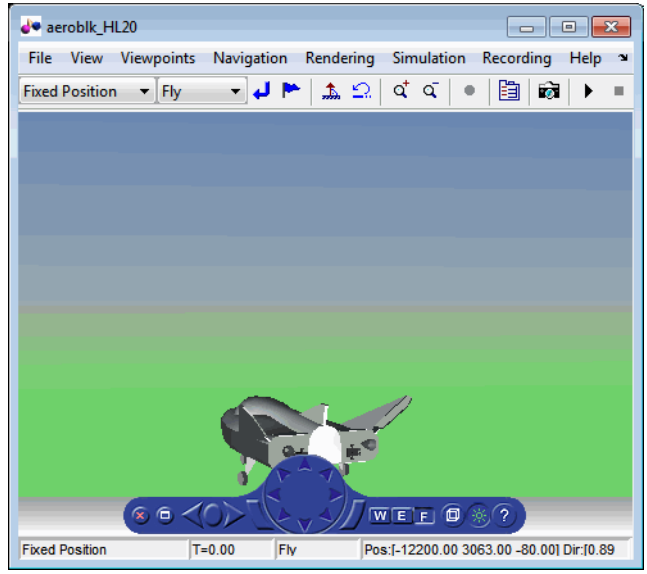
Modify the Model

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.

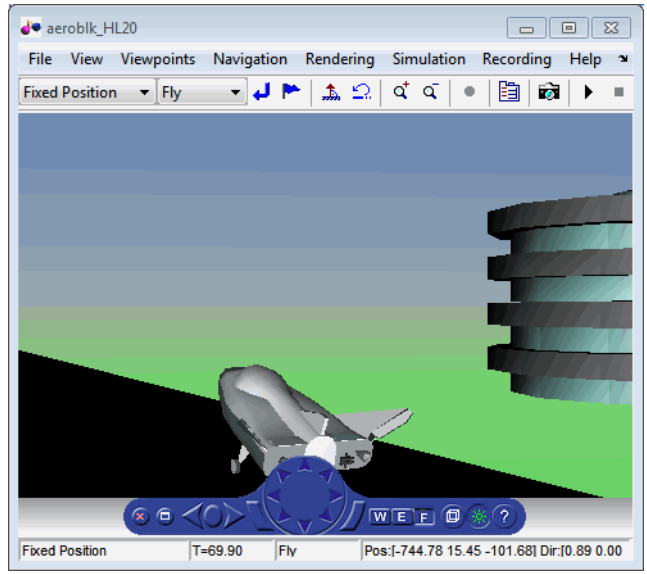
Change the Animation Point of View

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 the Help Browser:

1. Open the MATLAB Command Window.
2. Click the question mark.
3. Navigate to Aerospace Blockset and click the **Examples** tab.

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

- “Ideal Airspeed Correction” on page 3-2
- “1903 Wright Flyer” on page 3-9
- “NASA HL-20 Lifting Body Airframe” on page 3-18

Aerospace Blockset Software

- “Create Aerospace Models” on page 2-2
- “Build a Simple Actuator System” on page 2-4
- “About Aerospace Coordinate Systems” on page 2-10
- “Flight Simulator Interface” on page 2-19
- “Work with the Flight Simulator Interface” on page 2-24
- “Simulink Projects Template for Flight Simulation Applications” on page 2-43
- “Flight Instrument Gauges” on page 2-46
- “Display Measurements with Cockpit Instruments” on page 2-47
- “Calculate UT1 to UTC Values” on page 2-50

Create Aerospace Models

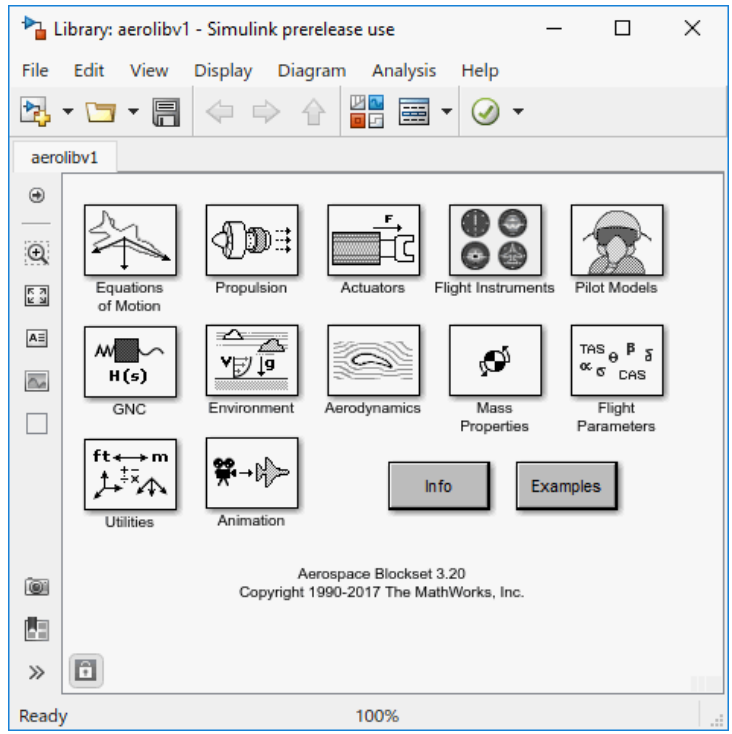
Basic Steps

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. You can access this library through the Simulink Library Browser or directly open the Aerospace Blockset window from the MATLAB command line:

```
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.

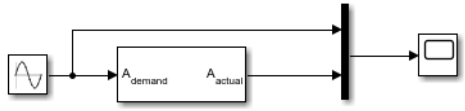
Build a Simple Actuator System

In this section...
“Build the Model” on page 2-4 “Run the Simulation” on page 2-8

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 2-5) explains how to build a model on Windows platforms. You can use this same procedure to build a model on Linux® platforms.

The section describes how to build the model. It does not describe how to set the configuration parameters for the model. See “Configuration Parameters Dialog Box Overview” (Simulink). That topic describes the Configuration Parameters Dialog Box 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

To create a new blank model and open the Simulink library browser:

1. On the MATLAB **Home** tab, click Simulink. In the Simulink start page, click the Blank Model template, and then click Create Model.
2. To open the Library Browser, click the browser button.
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.

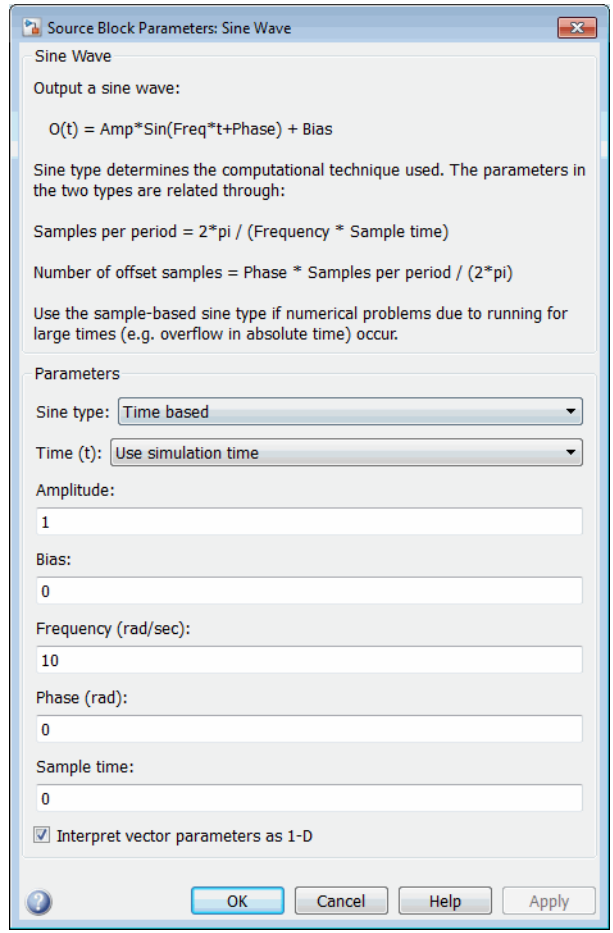
d Position the pointer near the first input port of the Mux block. Hold down the mouse button and drag the line that appears over the line from the output port of the Sine Wave block until double crosshairs appear. Release the mouse button. The lines are connected when a knot is present at their intersection.

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.

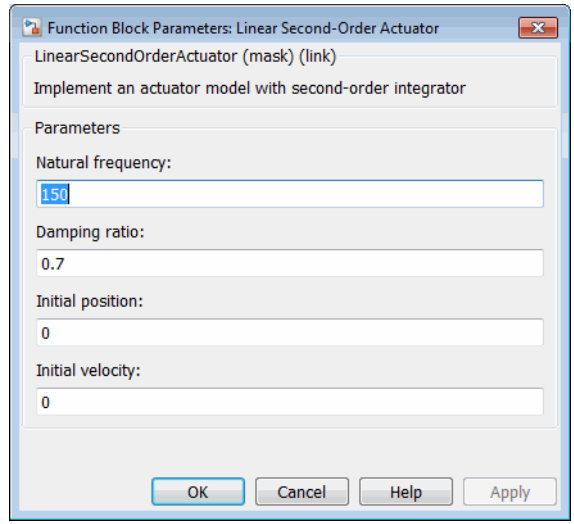
b Click **OK**.



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 “Run Simulations Programmatically” (Simulink).

About Aerospace Coordinate Systems

In this section...
“Fundamental Coordinate System Concepts” on page 2-10
“Coordinate Systems for Modeling” on page 2-11
“Coordinate Systems for Navigation” on page 2-14
“Coordinate Systems for Display” on page 2-16
“References” on page 2-18

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: force = mass × 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.

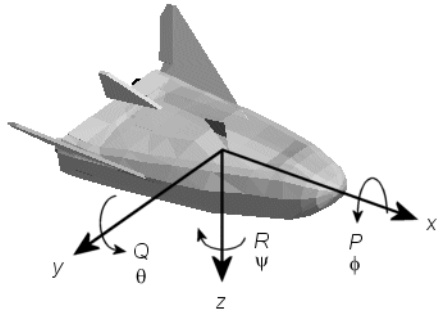
Translational Degrees of Freedom

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 Φ, Θ, Ψ. They are:

P or Φ	Roll about the *x*-axis
Q or Θ	Pitch about the *y*-axis
R or Ψ	Yaw about the *z*-axis



Unless otherwise specified, by default the software uses ZYX rotation order for Euler angles.

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.

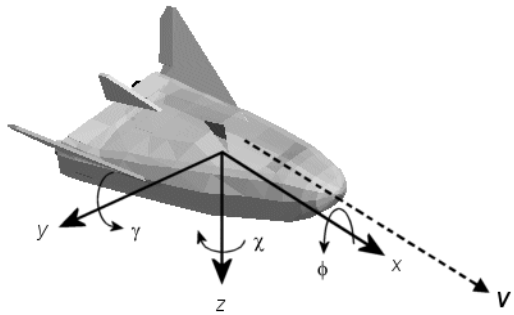
Translational Degrees of Freedom

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 Φ, γ, χ. They are:

Φ	Bank angle about the *x*-axis
γ	Flight path about the *y*-axis
χ	Heading angle about the *z*-axis



Unless otherwise specified, by default the software uses ZYX rotation order for Euler angles.

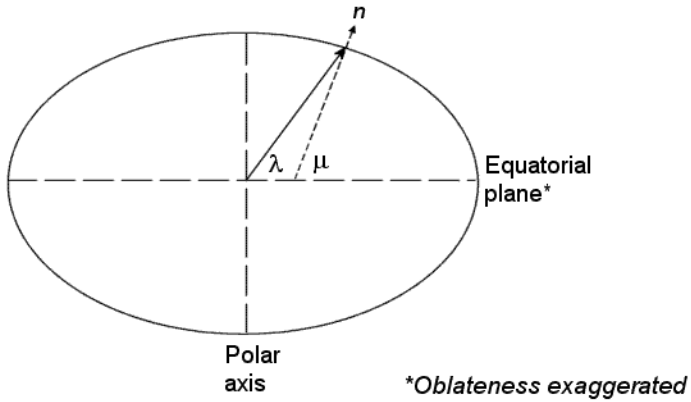
Coordinate Systems for Navigation

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* λ 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* μ on the Earth's surface is defined by the angle subtended by the surface normal vector n and the equatorial plane.

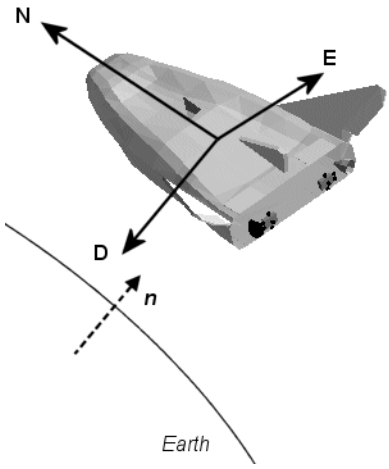


NED Coordinates

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***.

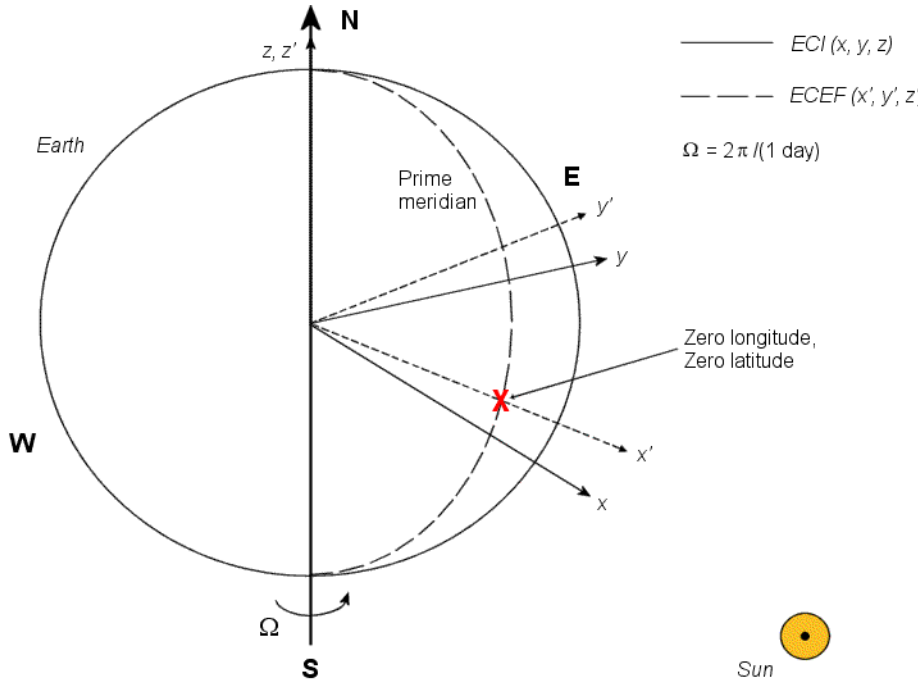
Flying at a constant altitude means flying at a constant z above the Earth's surface.



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 "Axes Appearance" (MATLAB) 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.

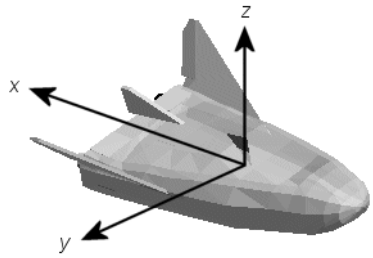
FlightGear Coordinates

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

- "Work with the Flight Simulator Interface" on page 2-24 discusses the blockset interface to FlightGear.
- See the FlightGear documentation at `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:

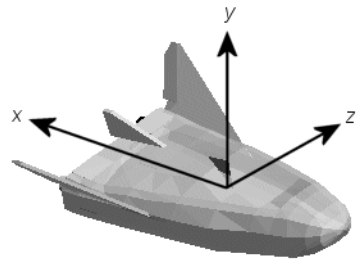
- The *x*-axis is positive toward the back of the vehicle.
- The *y*-axis is positive toward the right of the vehicle.
- The *z*-axis is positive upward, e.g., wheels typically have the lowest *z* values.



AC3D Coordinates

AC3D is a low-cost, widely used, geometry editor available from `www.ac3d.org`. 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. “Mapping Toolbox”.

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 2-19
“Supported FlightGear Versions” on page 2-19
“Obtain FlightGear” on page 2-20
“Configure Your Computer for FlightGear” on page 2-20
“FlightGear and Video Cards in Windows Systems” on page 2-21
“Install and Start FlightGear” on page 2-21
“Install Additional FlightGear Scenery” on page 2-23

About the FlightGear Interface

The Aerospace Blockset product supports an interface to the third-party FlightGear flight simulator, open-source software 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 the FlightGear 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 2-29 for interface details.

FlightGear is a separate software entity not created, owned, or maintained by MathWorks.

- To report bugs in or request enhancements to the Aerospace Blockset FlightGear interface, use the `form`.
- To report bugs or request enhancements to FlightGear itself, visit `FlightGear website`.

Supported FlightGear Versions

The Aerospace Blockset product supports these FlightGear versions:

- `v2.0`
- `v2.4`

- `v2.6`
- `v2.8`
- `v2.10`
- `v2.12`
- `v3.0`
- `v3.2`
- `v3.4`
- `v2016.1`
- `v2016.3`
- `v2017.1`
- `v2017.3`

If you are using a FlightGear version older than 2.0, update your FlightGear installation to a supported version. When you open the model, the software returns a warning or error. Obtain updated FlightGear software from `www.flightgear.org` in the download area.

Obtain FlightGear

You can obtain FlightGear from the FlightGear website 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 customizing and porting to custom environments.

Configure Your Computer for FlightGear

To use FlightGear, you must have a high-performance graphics card with stable drivers For more information, see the FlightGear CD distribution or the hardware requirements and documentation areas of the FlightGear website.

FlightGear performance and stability can be sensitive to computer video cards, driver versions, and driver settings. You need OpenGL® support with hardware acceleration activated. 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

For Windows systems, use the following graphics recommendations:

- A graphics card with acceptable OpenGL performance (as outlined at the FlightGear website).
- The latest tested and stable driver release for your video card. Test the driver thoroughly on a few computers before deploying to others.

For more information, see `FlightGear Hardware Recommendations`.

Setup on Linux, Macintosh, and Other Platforms

FlightGear distributions are available for Linux, Macintosh, and other platforms from the FlightGear website, `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 website.

FlightGear and Video Cards in Windows Systems

Your computer built-in video card, such as NVIDIA® cards, can conflict with FlightGear shaders. Consider one of these workarounds:

- In the device manager, disable the NVIDIA driver, enable the Windows system video driver, and start FlightGear. After starting FlightGear, you can reenable the NVIDIA video card.

  Note If you do not reenable the NVIDIA card, the Windows driver can affect shading quality, or cause your system to lose support for multiple monitors.

- Disable the FlightGear shaders by selecting the Generate Run Script block **Disable FlightGear shader options** check box.

Install and Start FlightGear

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

- Documentation section of the FlightGear website for installation instructions: `www.flightgear.org`.

- Hardware recommendations section of the FlightGear website.
- MATLAB system requirements.

Keep the following points in mind:

- Configure your computer graphics card before you install FlightGear. See the preceding section, “Configure Your Computer for FlightGear” on page 2-20.
- Shut down all running applications (including the MATLAB interface) before installing FlightGear.
- Install FlightGear in a folder path name composed of ASCII characters.
- MathWorks tests indicate that the operational stability of FlightGear is especially sensitive during startup. It is best not to 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 several platforms. This table lists the properties to consider before you start to use FlightGear.

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

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.

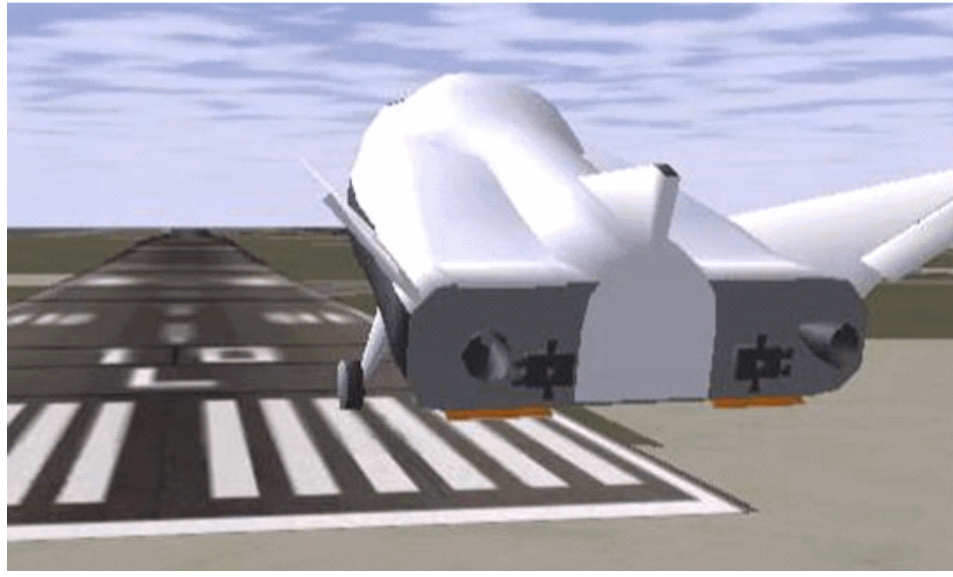
If you do not download scenery, you can direct FlightGear to download it automatically during simulation by selecting the Generate Run Script block **Install FlightGear scenery during simulation (requires Internet connection)** check box.

Work with the Flight Simulator Interface

In this section...
“Introduction” on page 2-24
“About Aircraft Geometry Models” on page 2-25
“Work with Aircraft Geometry Models” on page 2-27
“Run FlightGear with the Simulink Models” on page 2-29
“Run the HL-20 Example with FlightGear” on page 2-36
“Send and Receive Data” on page 2-38

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 2-19 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`. Another popular 3-D editor for aircraft models is Flight Sim Design Studio, distributed by Abacus Publications at `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` 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 2-21.

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 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 HL-20 Example with FlightGear” on page 2-36.

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 2-10 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 2-30 explains how to create the network connection you need.
- “Obtain the Destination IP Address” on page 2-30 starts by determining the IP address of the computer running FlightGear.
- “Send Simulink Data to FlightGear” on page 2-38 shows how to add and connect interface and pace blocks to your Simulink model.

- “Create a FlightGear Run Script” on page 2-31 shows how to write a FlightGear run script compatible with your Simulink model.
- “Start FlightGear” on page 2-33 guides you through the final steps to making the Simulink software work with FlightGear.
- “Improve Performance” on page 2-34 helps you speed your model up.
- “Run FlightGear and Simulink Software on Different Computers” on page 2-35 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

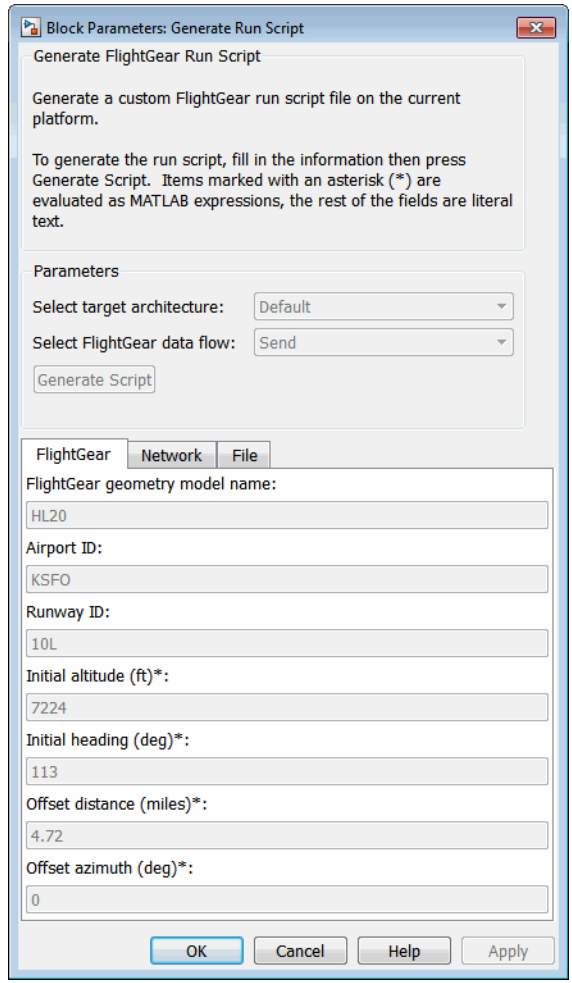
To start FlightGear with the desired initial conditions (location, date, time, weather, operating modes), it is best to create a run script by “Use the Generate Run Script Block” on page 2-31 or “Use the Interface Provided with FlightGear” on page 2-33.

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:

Platform	Extension
Windows	`.bat`
Linux and macOS	`.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-2017.3
>> SET FG_ROOT=D:\Applications\FlightGear-2017.3\data
>> cd \bin\
>> 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.

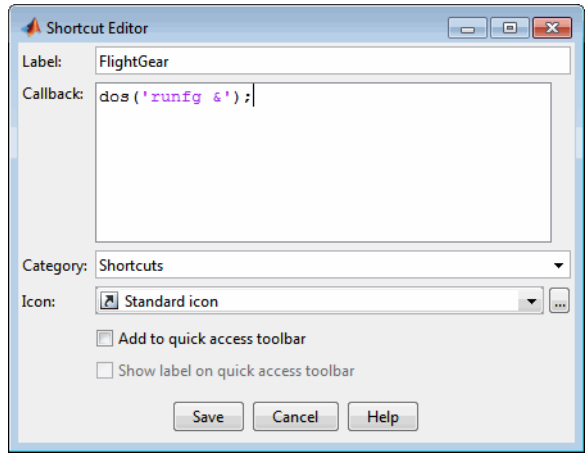
Start FlightGear

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 **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” (Simulink).)
- 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.

Tip If FlightGear uses more computer resources than you want, you can change its scheduling priority to a lesser one. For example, see commands like Windows `start` and Linux `nice` or their equivalents.

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 2-30.
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 2-38.
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 2-31.
4. Copy the generated run script to the target computer. Start FlightGear there. See “Start FlightGear” on page 2-33.
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.

Run the HL-20 Example with FlightGear

The Aerospace Blockset software contains an example model of the HL-20 lifting body that uses the FlightGear interface and Simulink Projects. This example illustrates many features of the Aerospace Blockset software. It also contains a Variant Subsystem block that you can use to specify the data source for the simulation. You might want to use the Variant Subsystem block to change the terrain data source or if you do not want to use FlightGear but still want to simulate the model.

To install and configure FlightGear before attempting to simulate this model, see “Flight Simulator Interface” on page 2-19. Also, before attempting to simulate this model, read “Install and Start FlightGear” on page 2-21.

Note Step 2 of this example copies the preconfigured geometries for the HL-20 simulation from *projectroot*\support to *FlightGear*\data\Aircraft\. It requires that you have system administrator privileges for your machine. If you do not have these privileges, manually copy these files, depending on your platform.

Windows

Copy the HL20 folder from *projectroot*\support folder to *FlightGear*\data\Aircraft\ folder. This folder contains the preconfigured geometries for the HL-20 simulation and HL20-set.xml. The file *projectroot*\support\HL20\Models\HL20.xml defines the geometry.

For Windows platforms, start the MATLAB app with administrator privileges. For example, in the Start menu, right click the MATLAB app, then select **Run as administrator**.

For more information, see “Import Aircraft Models into FlightGear” on page 2-27.

Linux

Copy the HL20 directory from *projectroot*/support directory to *$FlightGearBaseDirectory*/data/Aircraft/ directory. This directory contains the preconfigured geometries for the HL-20 simulation and HL20-set.xml. The file *projectroot*/support/HL20/Models/HL20.xml defines the geometry.

For more about this step, see “Import Aircraft Models into FlightGear” on page 2-27.

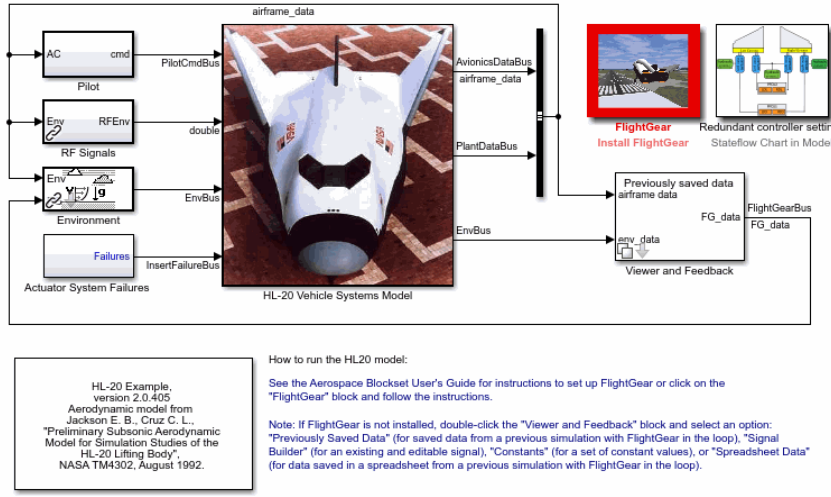
Mac

Copy the `HL20` folder from *`projectroot`*`/support` 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 *`projectroot`*`/support/HL20/Models/HL20.xml` defines the geometry.

For more about this step, see “Import Aircraft Models into FlightGear” on page 2-27.

---

1 Start the MATLAB interface. Open the example either by entering `asbhl20` in the MATLAB Command Window or by finding the example entry (HL-20 with FlightGear Interface) in the Aerospace Blockset Examples page. The project for the model starts and the model opens.



2 If this is your first time running FlightGear for this model, you need to create and run a customized FlightGear run script. You can do this with one of the following:

- In the model, double-click the Install FlightGear block and follow the steps in the block. Initially, this block is red. As you follow the steps outlined in the block, the block mask changes.

  To start FlightGear for the model, click **Launch HL20 in FlightGear**.

3 Now start the simulation and view the animation in FlightGear.

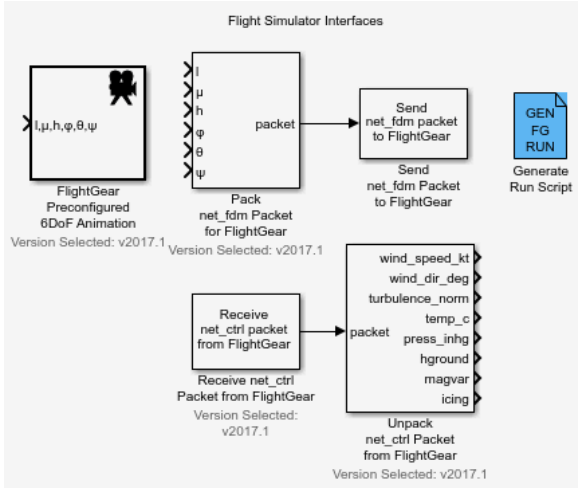
Note With the FlightGear window in focus, press the **V** key to alternate between the different aircraft views: cockpit view, helicopter view, chase view, and so on.

Send and Receive Data

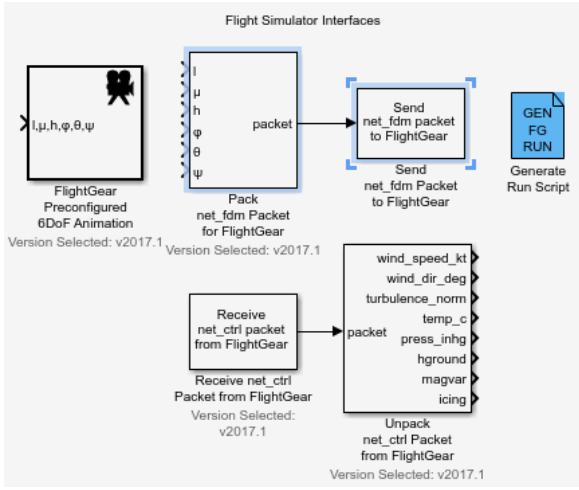
You can send and receive data between a Simulink model and a running FlightGear Flight Simulator.

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:

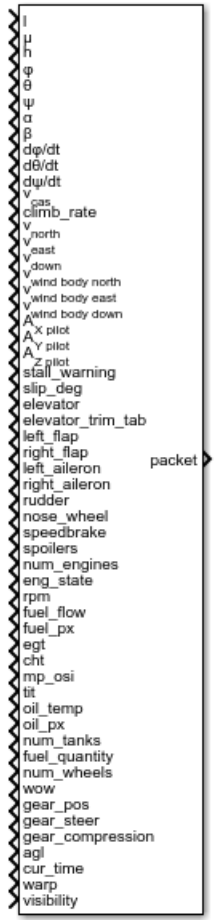


The FlightGear Preconfigured 6DoF Animation block is a subsystem containing the Pack net_fdm Packet for FlightGear and Send net_fdm Packet to FlightGear blocks:



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

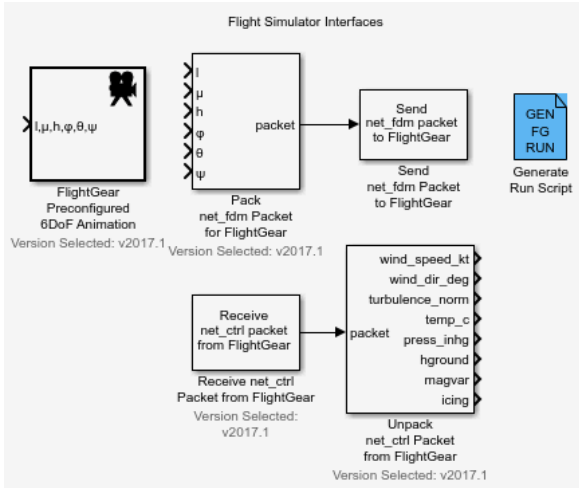
- The Pack net_fdm Packet for FlightGear block formats a binary structure compatible with FlightGear from model inputs. In the default configuration, the block displays only the 6DoF ports, but you can configure the full FlightGear interface supporting more than 50 distinct signals from the block dialog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



- 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 2-30.
- 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 `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 Version 2.6 or Later

On a Macintosh system with FlightGear v2.6 or later, 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.

Simulink Projects Template for Flight Simulation Applications

Flight Simulation Applications

Use Simulink Projects (Simulink) to help organize large flight simulation modeling projects and makes it easier to share projects with others.

The Aerospace Blockset software supplies a Simulink projects template that you can use to create your own flight simulation application. This template uses variant subsystems, model variants, and referenced models to implement flight simulation application components such as:

- An airframe that contains a 6DOF equation of motion environment model and actuator dynamics
- An inertial measurement unit (IMU) sensor model
- A visualization subsystem oriented for FlightGear
- A model of the nonlinear dynamics of the airframe
- A model of the linear dynamics of the airframe

Download the Flight Simulation Template

1. From Simulink, select **New > Simulink Project > Flight Simulation Example**.
2. In Simulink Project, in **Project name**, enter a project name, for example `FlightSimProj`.
3. In **Project folder**, enter a project folder or browse to the folder to contain the project, for example `FlightSimFolder`.
4. Click **Create**.

   If the folder does not exist, the dialog prompts you to create it. Click **Yes**.

   The software compiles the project, populates the project folders, and opens the main model, `flightSimulation`. All models and supporting files are in place for you to customize for your flight simulation application.

The flight simulation project template contains the following folders

- **controller**

  Contains the models for the Flight Control System (FCS) and its design. These models contain referenced models for different controller architectures needed for the design of aircraft simulation.

- **libraries**

  Contains the libraries used by the models.

- **linearAirframe**

  Contains the linear dynamics of the airframe.

- **mainModels**

  Contains the top-level simulation model, `flightSimulation`.

- **nonlinearAirframe**

  Contains a model of the nonlinear dynamics of the airframe.

- **src**

  Contains C code. For simulation, it also has two folders that contain S-functions for simulation.

- **tasks**

  Contains scripts to run the model. These scripts do not run continuously during the simulation process.

- **tests**

  Contains a sample test harness, including a test file, `linearTest`, and a script, `runProjectTests`, that runs the designated tests in the project.

- **utilities**

  Contains project-specific maintenance task utilities.

- **work**

  Contains files generated from every run. These files derive from source files, such as the MEX-file that you build from S-function C code.

In Shortcuts, Simulink projects creates shortcuts for common tasks such as:

- Set up and clean up projects.
- Add projects to MATLAB paths.
- Initialize variables.
- Rebuild S-functions.
- Run project tests.
- Start the top model.

Add Airframe Dynamics and Controller Algorithm to the Project

1. To open the `linearAirframe` model, in `flightSimulation` double-click the Airframe subsystem.
2. Double-click the Nonlinear subsystem.
3. In the AC model, add your airframe dynamics.
4. Save the model.

Add Controller Algorithm to the Project

1. To open the `flightControlSystem` model, in `flightSimulation`, double-click the FCS subsystem.
2. In the Controller subsystem, add your controller algorithm.
3. Save the model.

Other things to try:

- Simulate your model.
- Explore the **tests** folder for sample tests for your application.

See Also

Related Examples

- “Create a New Project Using Templates” (Simulink)
- Quadcopter Project

Flight Instrument Gauges

Use the blocks for flight instrument gauges to visualize navigation variables, such as altitude and heading. These blocks, located in the Flight Instruments library, represent standard cockpit instruments:

- Airspeed Indicator
- Altimeter
- Artificial Horizon
- Climb Rate Indicator
- Exhaust Gas Temperature (EGT) Indicator
- Heading Indicator
- Revolutions Per Minute (RPM) Indicator
- Turn Coordinator

See Also

Airspeed Indicator | Altimeter | Artificial Horizon | Climb Rate Indicator | Exhaust Gas Temperature (EGT) Indicator | Heading Indicator | Revolutions Per Minute (RPM) Indicator | Turn Coordinator

Related Examples

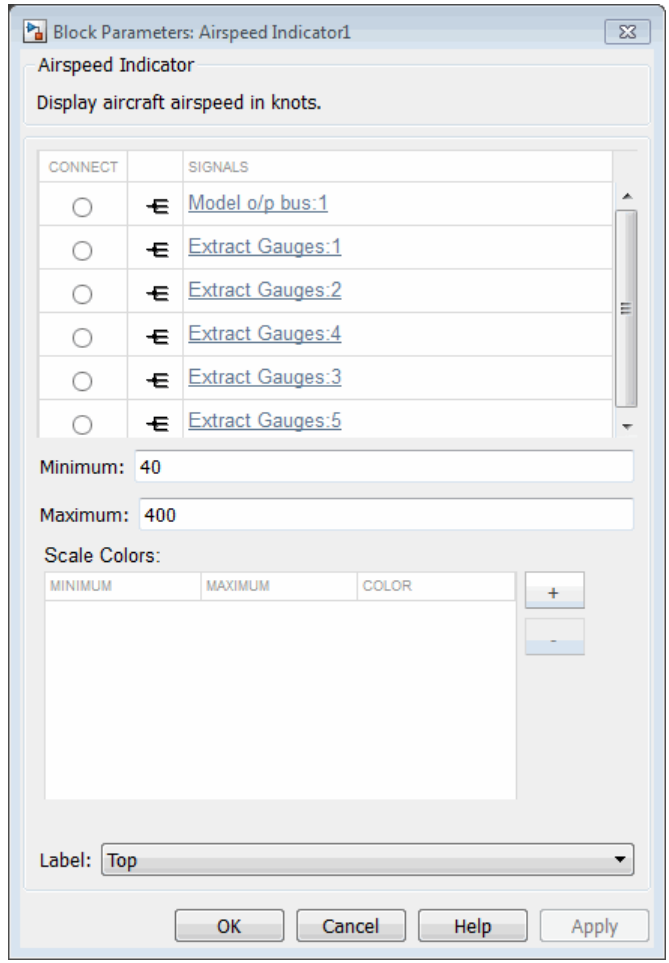
- “Display Measurements with Cockpit Instruments” on page 2-47

Display Measurements with Cockpit Instruments

You can view signal data using any of the Flight Instrument blocks. This example uses the HL-20 Lifting Body and Controller with Flight Instrumentation Blocks model. In this example, connect a gauge so that you can view the aircraft heading.

1. To open the model, at the MATLAB command window, enter `aeroblk_HL20_Gauges`.
2. Open the Visualization subsystem.

   There is an existing Airspeed Indicator block in the model.
3. Add a second Airspeed Indicator block from the Flight Instruments library to the subsystem.
4. Open the new Airspeed Indicator block.
5. Select the Extract Flight Instruments block.
6. In the new Airspeed Indicator block, observe that the block connection table is filled with signals from the Extract Flight Instruments block that you can observe.



7 Select the option button next to `Extract_Gauges:2` in the connection table.

8 To connect the `Extract_Gauges:2` signal to the Airspeed Indicator block, click **OK**.

9 Simulate the model and observe the gauge as it registers the data.

10 Close the model without saving it.

See Also

Airspeed Indicator | Altimeter | Artificial Horizon | Climb Rate Indicator | Exhaust Gas Temperature (EGT) Indicator | Heading Indicator | Revolutions Per Minute (RPM) Indicator | Turn Coordinator

More About

- “Flight Instrument Gauges” on page 2-46

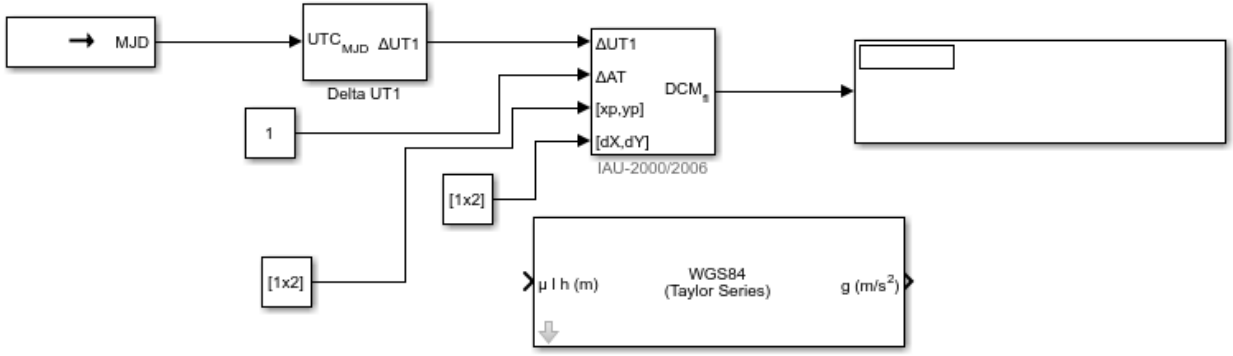
Calculate UT1 to UTC Values

Calculate the difference between principal Universal Time (UT1) and Coordinated Universal Time (UTC) according to International Earth Rotation Service (IERS) by using the Delta UT1 block. Use the Delta UT1 block with these axes transformation blocks:

- LLA to ECI Position
- ECI Position to LLA
- Direction Cosine Matrix ECI to ECEF
- ECI Position to AER

To calculate the difference between UT1 and UTC, the Delta UT1 block requires the modified Julian date. This example uses the Julian Date Conversion block. However, you can calculate the modified Julian data with other methods. For example, you can use the mjuliandate`mjuliandate` function from the Aerospace Toolbox software to calculate the date and input the result to the Delta UT1 block.

Use the Delta UT1 Block to Create Difference Values for the Direction Cosine Matrix ECI to ECEF Block



This model shows how a Direction Cosine Matrix ECI to ECEF block uses the output from the Delta UT1 and Julian Data Conversion blocks to obtain the difference between UTC and Universal Time (UT1).

1 Drag these blocks into a new model and connect them as shown:

- Julian Date Conversion
- Delta UT1
- Direction Cosine Matrix ECI to ECEF
- Display
- Three Constant blocks

2. Set up the Julian Date Conversion to convert the date December 28, 2015 to its modified Julian date equivalent. This date must match the one specified in the Direction Cosine Matrix ECI to ECEF.

   - For **Year**, enter `2015`.
   - For **Month**, enter `12`.
   - For **Day**, enter `28`.
   - To calculate the modified Julian date for December 28, 2015, select the **Calculate modified Julian date** check box.
   - For **Time increment**, select `None`.

3. Leave the default settings for Delta UT1. By default, the block calculates the difference between Universal Time (UT1) and Universal Coordinated Time (UTC) to using the `aeroiersdata.mat` file supplied with the Aerospace Blocksetsoftware.

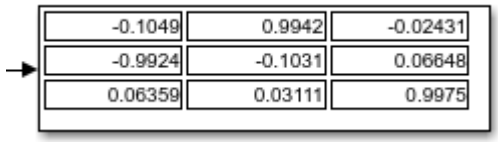
4. Set up the Direction Cosine Matrix ECI to ECEF block to work with the Universal Coordinated Time (UTC) December 28, 2015. This date must match the one specified in the Julian Date Conversion block:

   - For **Year**, enter `2015`.
   - For **Month**, enter `12`.
   - For **Day**, enter `28`.
   - For **Time increment**, select `None`.

5. Set up the $\Delta UT1$, ΔAT, and polar displacement of the Earth inputs for the Direction Cosine Matrix ECI to ECEF.

   - Constant — Set **Constant value** to `1`.
   - Constant1 — Set **Constant value** to `l`.
   - Constant2 — Set **Constant value** to `[.05 .05]`.

6. Save and run the model. Observe the resulting direction cosine matrix in the Display block.



See Also

Delta UT1 | Direction Cosine Matrix ECI to ECEF | Julian Date Conversion

Case Studies

- “Ideal Airspeed Correction” on page 3-2
- “1903 Wright Flyer” on page 3-9
- “NASA HL-20 Lifting Body Airframe” on page 3-18

Ideal Airspeed Correction

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

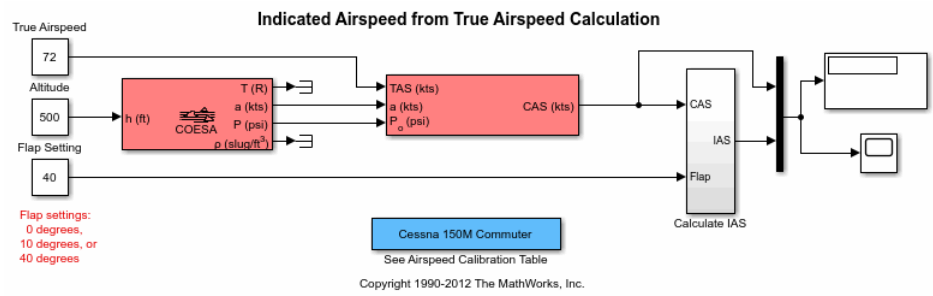
Introduction

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

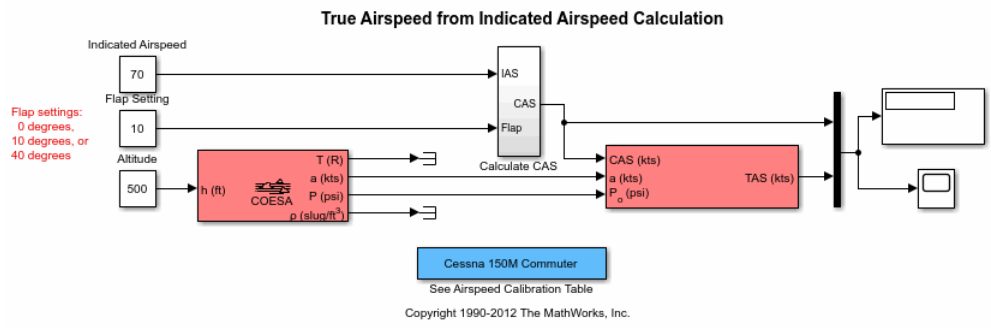
Airspeed Correction Models

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 3-4 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.

Airspeed Input	Airspeed Output
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 3-5
- “Calibrated Airspeed Implementation” on page 3-6
- “Equivalent Airspeed Implementation” on page 3-6

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

α	Speed of sound at altitude in m/s
δ	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

ρ_0	Air density at mean sea level in kg/m^3
P_0	Static pressure at mean sea level in N/m^2
γ	Ratio of specific heats
q	Dynamic pressure at mean sea level in N/m^2

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:

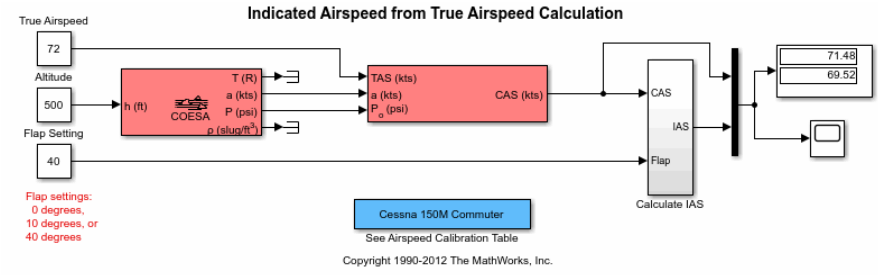
ρ_0	Air density at mean sea level in kg/m^3
P	Static pressure at altitude in N/m^2
γ	Ratio of specific heats

q	Dynamic pressure at mean sea level in N/m^2

Simulate Airspeed Correction

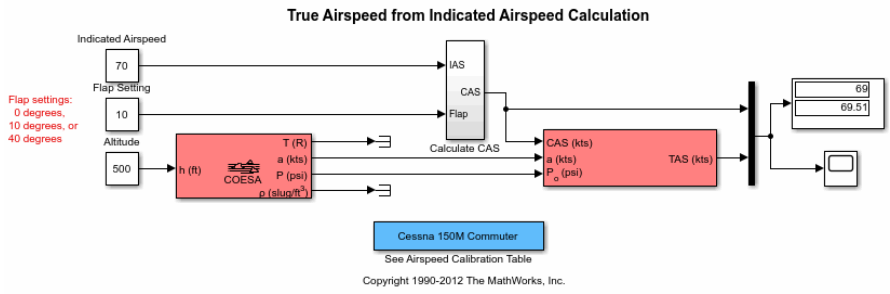
In the `aeroblk_indicated` model, the aircraft is defined to be traveling at a constant speed of 72 knots (true airspeed) and altitude of 500 feet. The flaps are set to 40 degrees. The COESA Atmosphere Model block takes the altitude as input and outputs the speed of sound and air pressure. Taking the 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.


True Airspeed from Indicated Airspeed Calculation
Indicated Airspeed
70
Flap Setting
10
Flap settings:
0 degrees,
10 degrees, or
40 degrees
Altitude
500
h (ft)
COESA
T (R)
a (kts)
P (psi)
ρ (slug/ft^3)
IAS
CAS
Flap
Calculate CAS
CAS (kts)
a (kts)
P_o (psi)
TAS (kts)
69
69.51
Cessna 150M Commuter
See Airspeed Calibration Table
Copyright 1990-2012 The MathWorks, Inc.

1903 Wright Flyer

In this section...
“Introduction” on page 3-9
“Wright Flyer Model” on page 3-10
“Airframe Subsystem” on page 3-10
“Environment Subsystem” on page 3-13
“Pilot Subsystem” on page 3-14
“Run the Simulation” on page 3-15
“References” on page 3-16

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].

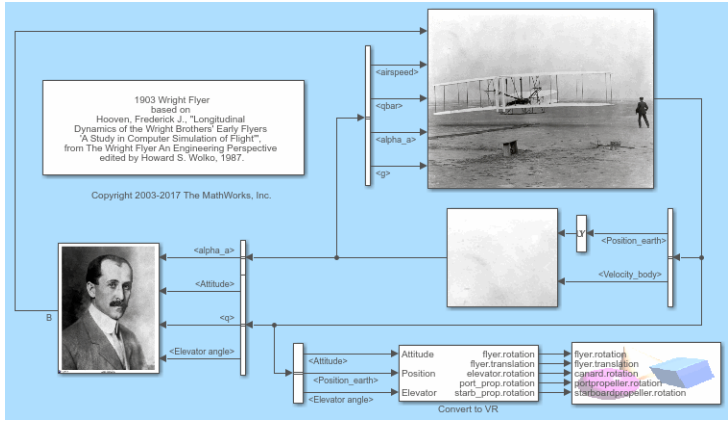
Under these constraints, the model recreates the longitudinal flight dynamics that pilots of the Wright Flyer would have experienced. Because they were able to control lateral motion, Orville and Wilbur Wright were able to maintain a relatively straight flight path.

Note, running this model generates information messages in the MATLAB Command Window and assertion warning messages in the Diagnostic Viewer. This is because the

model illustrates the use of the Assertion block to indicate that the flyer is hitting the ground when landing.

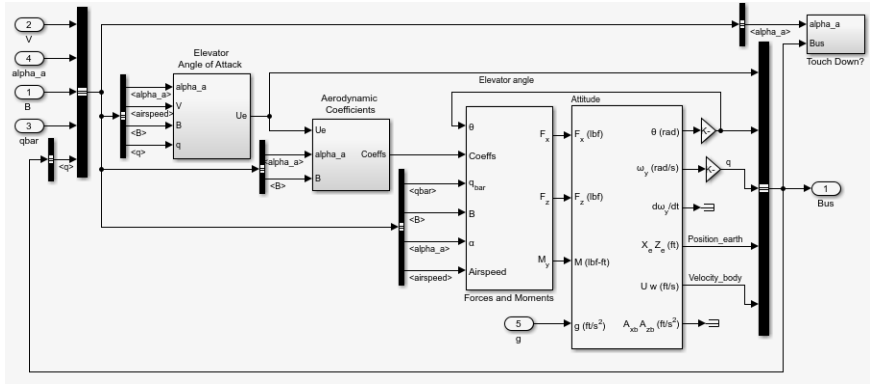
Wright Flyer Model

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.

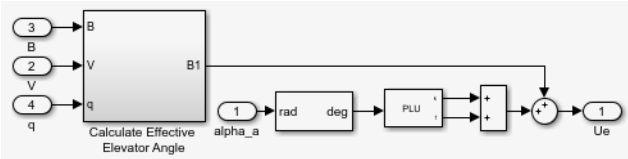


The Airframe subsystem consists of the following parts:

- “Elevator Angle of Attack Subsystem” on page 3-11
- “Aerodynamic Coefficients Subsystem” on page 3-11
- “Forces and Moments Subsystem” on page 3-11
- “3DOF (Body Axes) Block” on page 3-12

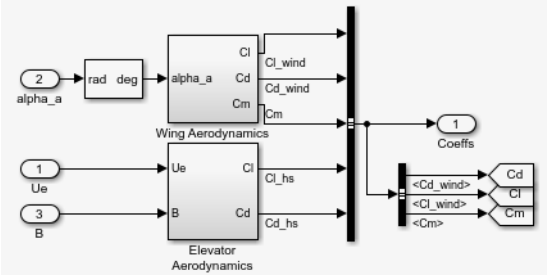
Elevator Angle of Attack Subsystem

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

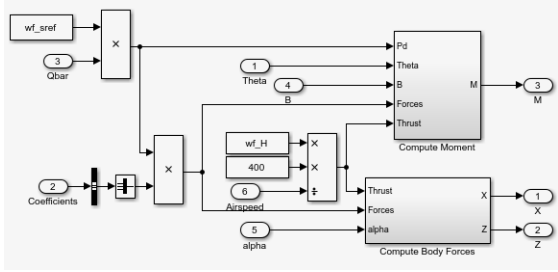
The Aerodynamic Coefficients subsystem contains aerodynamic data and equations for calculating the aerodynamic coefficients, which are summed and passed to the Forces and Moments subsystem. Stored in data sets, the aerodynamic coefficients are determined by interpolation using Prelookup blocks.



Forces and Moments Subsystem

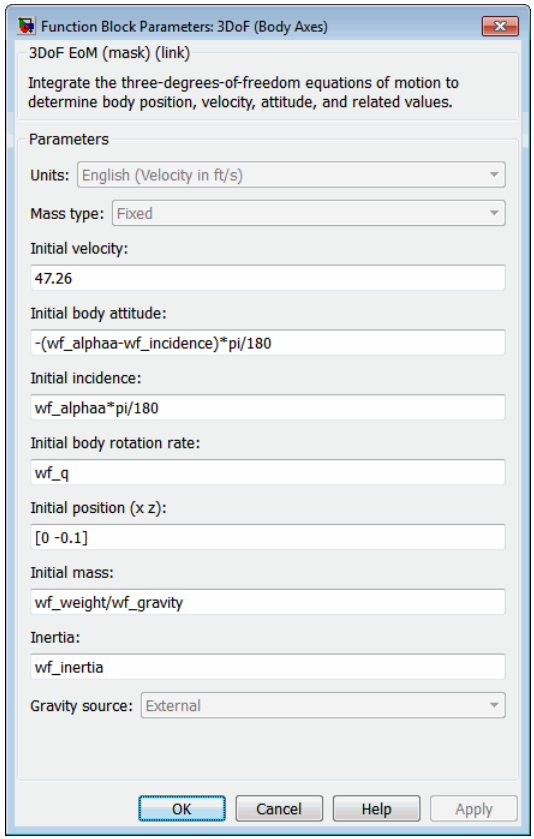
The aerodynamic forces and moments acting on the airframe are generated from aerodynamic coefficients. The Forces and Moments subsystem calculates the body forces and body moments acting on the airframe about the center of gravity. These forces and

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

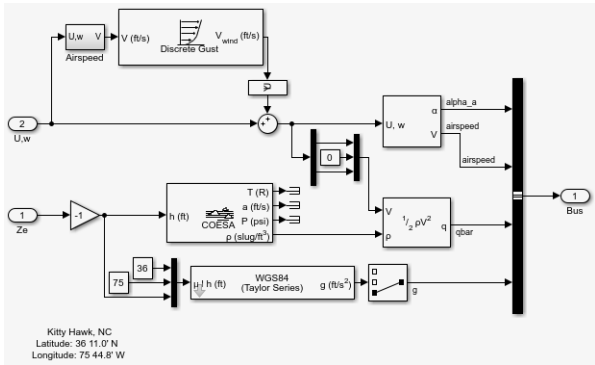
Environment Subsystem

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

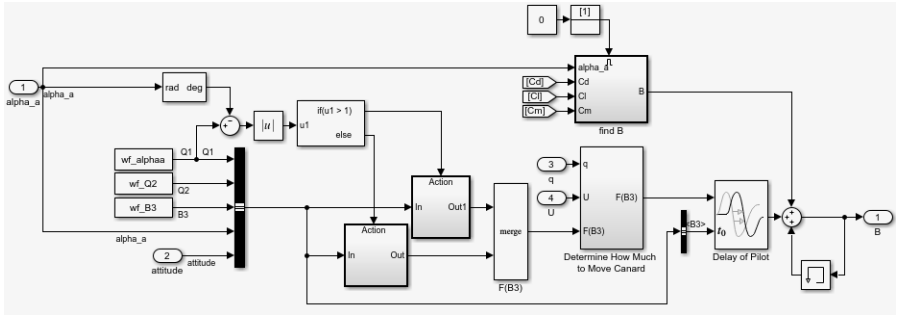
- The Incidence & Airspeed block calculates the angle of attack and airspeed.
- The COESA Atmosphere Model block calculates the air density.
- The Dynamic Pressure block computes the dynamic pressure from the air density and velocity.
- The WGS84 Gravity Model block produces the gravity at the Wright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



Pilot Subsystem

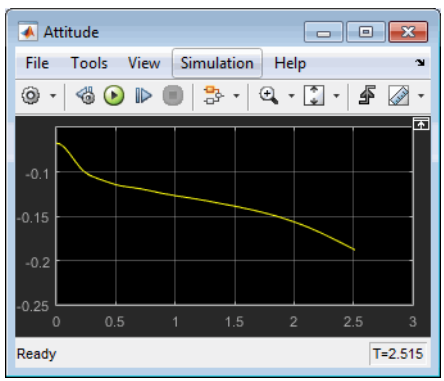
The Pilot subsystem controls the aircraft by responding to both pitch angle (attitude) and angle of attack. If the angle of attack differs from the set angle of attack by more than one degree, the Pilot subsystem responds with a correction of the elevator (canard) angle. When the angular velocity exceeds +/- 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_alphaa`) 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.



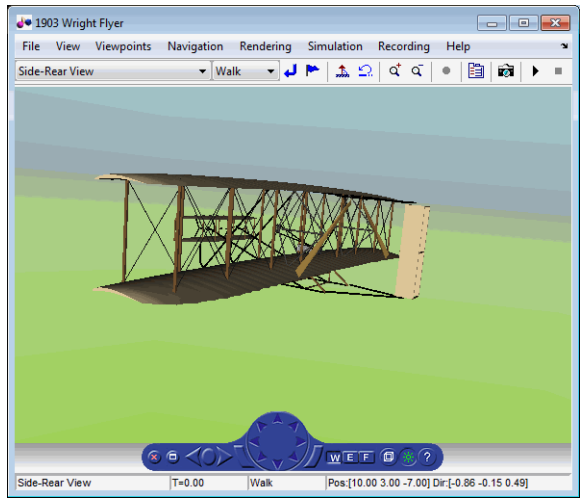
Attitude Scope (Measured in Radians)

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 3-18
“NASA HL-20 Lifting Body” on page 3-18
“The HL-20 Airframe and Controller Model” on page 3-19
“References” on page 3-28

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.

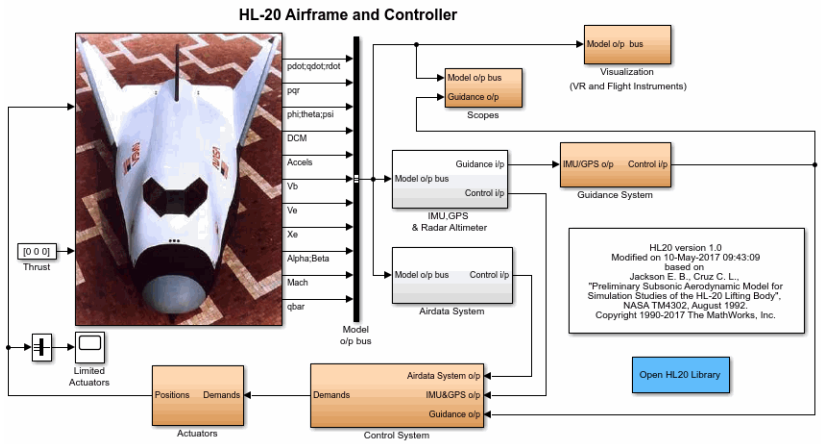
Potential uses for the HL-20 include

- 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:

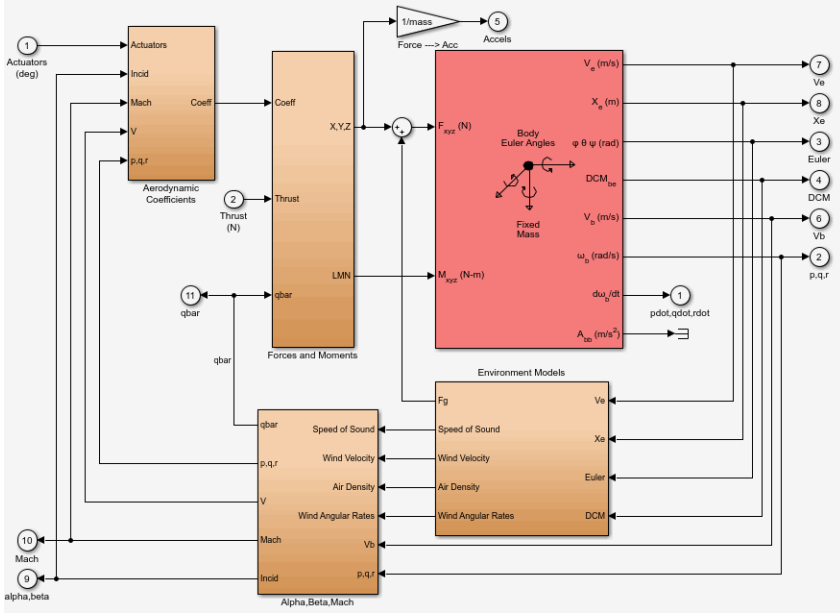
- The airframe is assumed to be rigid and have constant mass, center of gravity, and inertia, since the model represents only the unpowered reentry portion of a mission.
- HL-20 is assumed to be a laterally symmetric vehicle.
- Compressibility (Mach) effects are assumed to be negligible.
- Control effectiveness 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 3-21
- “Environmental Models Subsystem” on page 3-21
- “Alpha, Beta, Mach Subsystem” on page 3-24
- “Aerodynamic Coefficients Subsystem” on page 3-24
- “Forces and Moments Subsystem” on page 3-28



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.

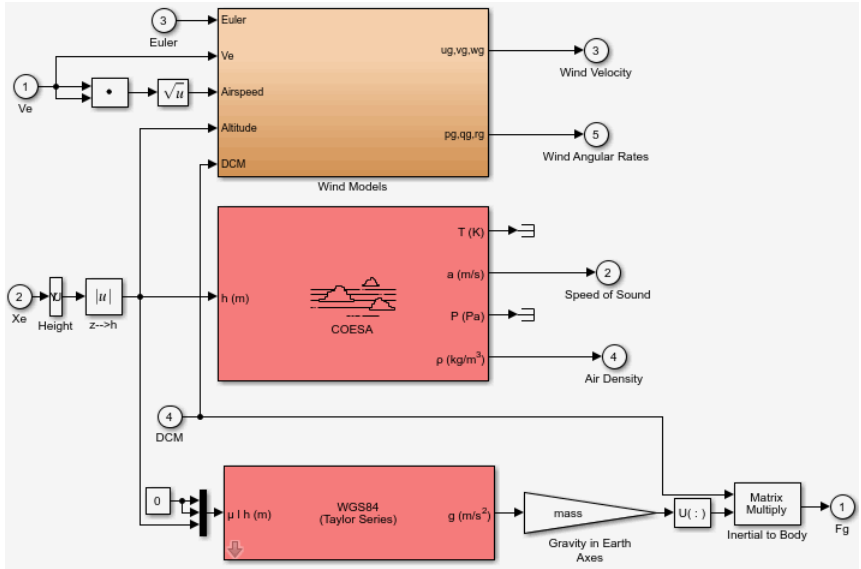
  See the COESA Atmosphere Model block reference page for more information on this block.
- The Wind Models subsystem contains the following blocks:
  - The Wind Shear Model block adds wind shear to the model.

    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.

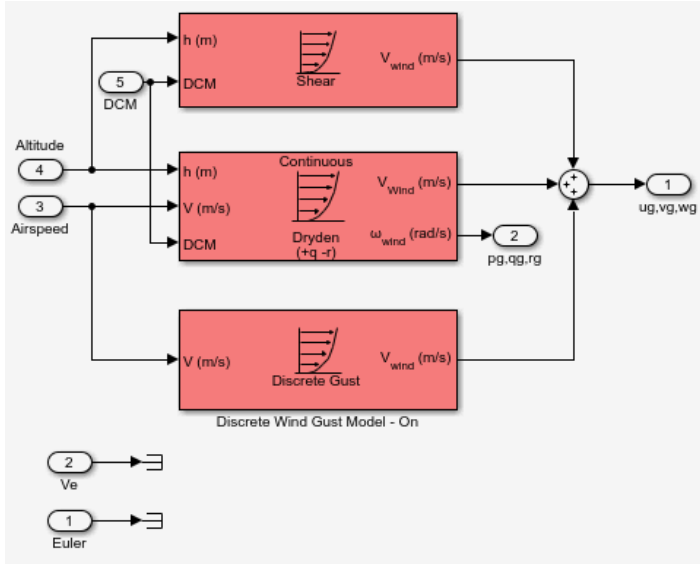
    See the Discrete Wind Gust Model block reference page for more information on this block.
  - The Dryden Wind Turbulence Model (Continuous) block uses the Dryden spectral representation to add turbulence to the aerospace model by passing band-limited white noise through appropriate forming filters.

    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



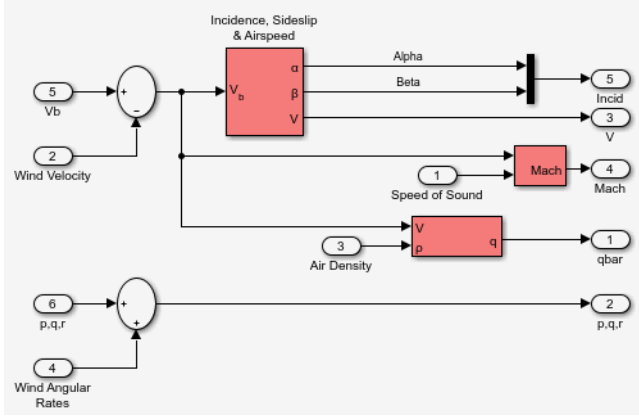
Wind Models in HL-20 Airframe Model

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 (α, β)
- 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

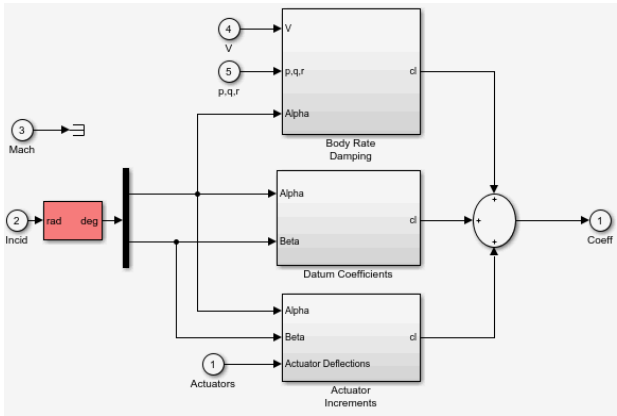
The Aerodynamic Coefficients subsystem contains aerodynamic data and equations for calculating the six aerodynamic coefficients, which are implemented as in reference [1]. The six aerodynamic coefficients follow.

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_hl20.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.

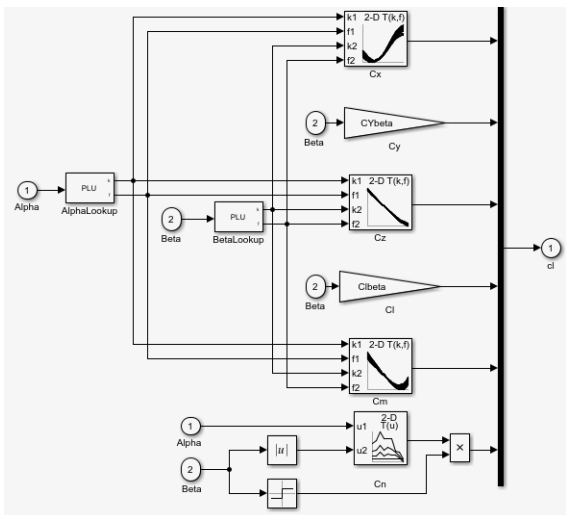
The equations for calculating the six aerodynamic coefficients are divided among three subsystems:

- “Datum Coefficients Subsystem” on page 3-26
- “Body Rate Damping Subsystem” on page 3-26
- “Actuator Increment Subsystem” on page 3-27

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].

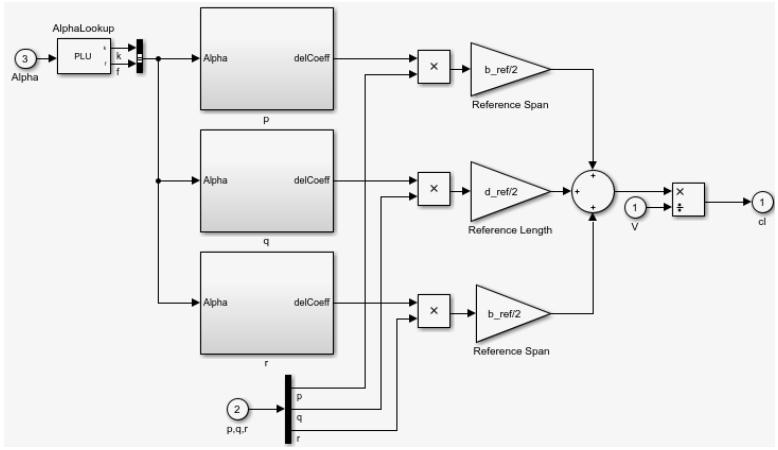
Datum Coefficients Subsystem

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.



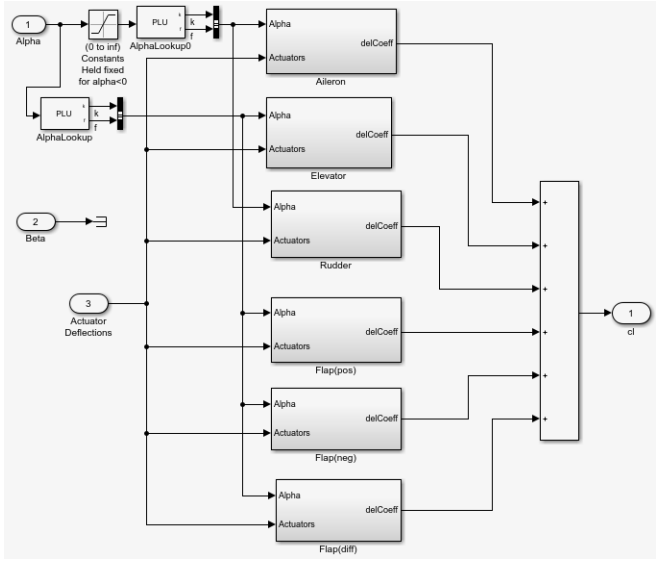
Body Rate Damping Subsystem

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



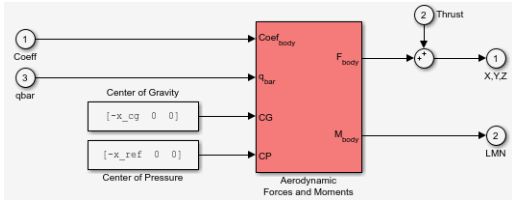
Actuator Increment Subsystem

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] Morring, 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/h/hl-20.html

Blocks — Alphabetical List

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

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.

Parameters

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. 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*. 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. In the case of 1-D scheduling, the *C*-matrix should have three dimensions, the last one corresponding to the scheduling variable *v*. 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*. 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_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	Any	Contains the measurements.
Second		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 (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)]

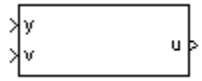
2D Controller [A(v),B(v),C(v),D(v)]

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

Introduced before R2006a

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

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

$$\begin{aligned}
\dot{x}_1 &= A_1 x_1 + B_1 y \\
u_1 &= C_1 x_1 + D_1 y \\
\dot{x}_2 &= A_2 x_2 + B_2 y \\
u_2 &= C_2 x_2 + D_2 y \\
u &= (1-\lambda) u_1 + \lambda u_2
\end{aligned}$$

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

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

Parameters

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 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

Input	Dimension Type	Description
First	Any	Contains the measurements.
Second		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 [A(v),B(v),C(v),D(v)]

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

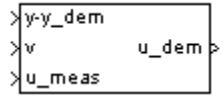
1D Self-Conditioned [A(v),B(v),C(v),D(v)]

2D Controller Blend

Introduced before R2006a

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

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.

Parameters

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];`.

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*.

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)]

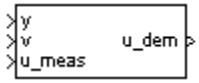
2D Observer Form [A(v),B(v),C(v),F(v),H(v)]

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

Introduced before R2006a

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

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.

Parameters

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];`.

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.

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)]

Introduced before R2006a

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

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

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

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.

Parameters

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 $v1$ 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 *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*.

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.

Input	Dimension Type	Description
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 `aeroblk_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)]

Introduced before R2006a

2D Controller Blend

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.

Parameters

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 *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];`.

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 *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*.

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)]

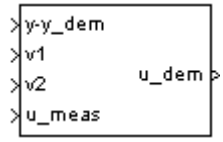
2D Observer Form [A(v),B(v),C(v),F(v),H(v)]

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

Introduced before R2006a

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

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:

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

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.

Parameters

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];`.

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];`.

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 *v*1 and *v*2. Hence, for example, if the *F*-matrix corresponding to the first entry of *v*1 and first entry of *v*2 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 *v*1 and *v*2. Hence, for example, if the *H*-matrix corresponding to the first entry of *v*1 and first entry of *v*2 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 *v*1 should be same as the size of the third dimension of *A*, *B*, *C*, *F*, and *H*.

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*, *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*.

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.

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

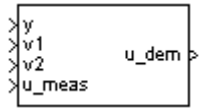
2D Self-Conditioned [A(v),B(v),C(v),D(v)]

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

Introduced before R2006a

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

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

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

in the self-conditioned form

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

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 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.

Parameters

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];`.

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*.

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.

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

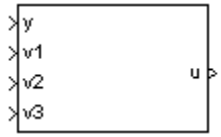
2D Observer Form [A(v),B(v),C(v),F(v),H(v)]

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

Introduced before R2006a

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

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

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

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.

Parameters

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];`.

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)]

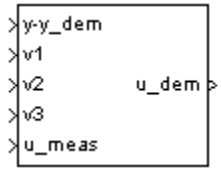
3D Observer Form [A(v),B(v),C(v),F(v),H(v)]

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

Introduced before R2006a

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

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.

Parameters

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];`.

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*, *F*, and *H*.

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*, *F*, and *H*.

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*, *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*.

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.

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)]

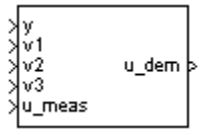
3D Controller [A(v),B(v),C(v),D(v)]

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

Introduced before R2006a

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

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

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

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.

Parameters

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];`.

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*.

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.

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)]

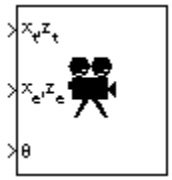
3D Controller [A(v),B(v),C(v),D(v)]

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

Introduced before R2006a

3DoF Animation

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.

Parameters

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.

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	1-by-1 scalar	Contains the attitude of the craft in radians.

Examples

See `aero_guidance` for an example of this block.

See Also

6DoF Animation

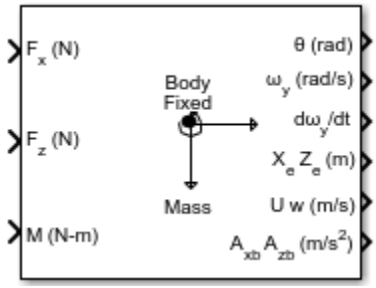
FlightGear Preconfigured 6DoF Animation

The figure axes properties CameraPosition and CameraViewAngle

Introduced before R2006a

3DOF (Body Axes)

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

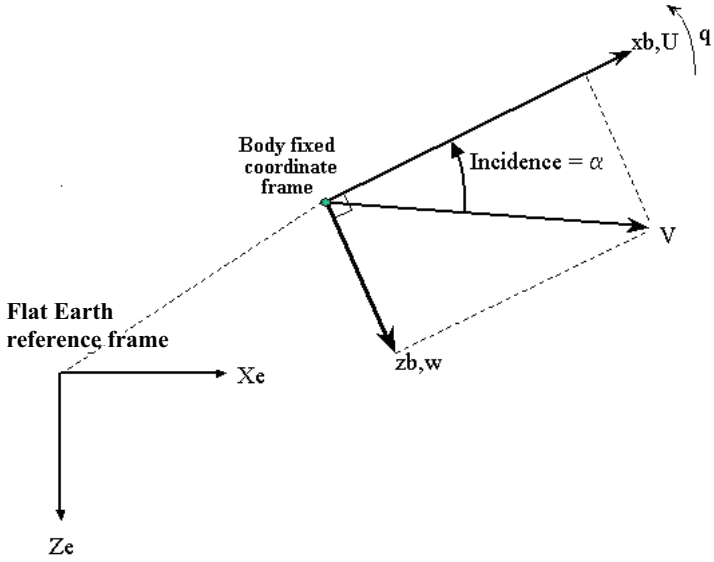


Library

Equations of Motion/3DOF

Description

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



The equations of motion are

$$A_{xb} = \dot{u} = \frac{F_x}{m} - qw - g\sin\theta, A_{xe} = \frac{F_x}{m} - \varepsilon\sin\theta$$

$$A_{zb} = \dot{w} = \frac{F_z}{m} + qu + g\cos\theta, A_{ze} = \frac{F_z}{m} + \varepsilon\cos\theta$$

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

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

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

Parameters

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 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, (θ_0).

Initial incidence

A scalar value for the initial angle between the velocity vector and the body, (α_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 flat Earth 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.

Include inertial acceleration

Select this check box to enable an additional output port for the accelerations in body-fixed axes with respect to the inertial frame. You typically connect this signal to the accelerometer.

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.

Output	Dimension Type	Description
First		Contains the pitch attitude, within ±pi, in radians (θ).

Output	Dimension Type	Description
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 flat Earth reference frame, (Xe, Ze).
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, (Ax, Az).
Seventh	Two-element vector	Contains the accelerations in body-fixed axes with respect to inertial frame (flat Earth). You typically connect this signal to the accelerometer.

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)

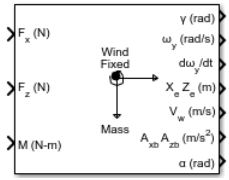
Simple Variable Mass 3DOF (Body Axes)

Simple Variable Mass 3DOF (Wind Axes)

Introduced in R2006a

3DOF (Wind Axes)

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

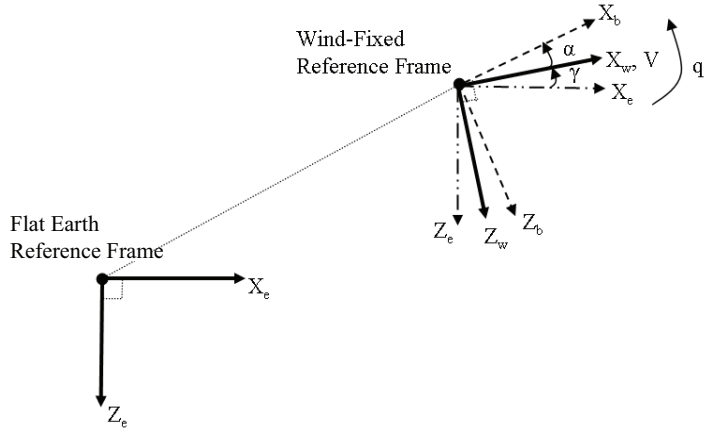


Library

Equations of Motion/3DOF

Description

The 3DOF (Wind Axes) block considers the rotation in the vertical plane of a wind-fixed coordinate frame about a flat Earth reference frame.



The equations of motion are

$$\dot{V} = \frac{F_{x_{wind}}}{m} - g\sin\gamma$$

$$\dot{\alpha} = \frac{F_{z_{wind}}}{mV\cos\beta} + q + \frac{g}{V\cos\beta}\cos\gamma$$

$$\dot{q} = \dot{\theta} = \frac{M_{y_{body}}}{I_{yy}}$$

$$\dot{\gamma} = q - \dot{\alpha}$$

$$A_{be} = \begin{bmatrix} A_{xe} \\ A_{ze} \end{bmatrix} = DCM_{wb}\left[\frac{F_w}{m} - g\sin\gamma\right]$$

$$A_{bb} = \begin{bmatrix} \dot{u}_b \\ \dot{w}_b \end{bmatrix} = DCM_{wb}\left[\frac{F_w}{m} - g\sin\gamma - \overline{\omega}_w \times \overline{V}_w\right]$$

$$\overline{F}_w = \begin{bmatrix} F_{x_{wind}} \\ F_{z_{wind}} \end{bmatrix}, \overline{V}_w = \begin{bmatrix} V_{x_{wind}} \\ V_{z_{wind}} \end{bmatrix}, \overline{\omega}_w = q$$

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

Parameters

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

Units	Forces	Moment	Acceleration	Velocity	Position	Mass	Inertia
`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, (γ_0).

Initial incidence

A scalar value for the initial angle between the velocity vector and the body, (α_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 flat Earth 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.

Include inertial acceleration

Select this check box to enable an additional output port for the accelerations in body-fixed axes with respect to the inertial frame. You typically connect this signal to the accelerometer.

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, within ±pi, in radians (γ).
Second		Contains the pitch angular rate, in radians per second (ω_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 flat Earth reference frame, (Xe, Ze).
Fifth	Two-element vector	Contains the velocity of the body resolved into the wind-fixed coordinate frame, (V, 0).

Output	Dimension Type	Description
Sixth	Two-element vector	Contains the acceleration of the body resolved into the body-fixed coordinate frame, (*Ax*, *Az*).
Seventh	Scalar	Contains the angle of attack, (α).
Eighth	Two-element vector	Contains the accelerations in body-fixed axes with respect to inertial frame (flat Earth). You typically connect this signal to the accelerometer.

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)

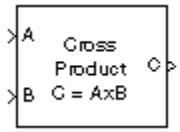
Simple Variable Mass 3DOF (Body Axes)

Simple Variable Mass 3DOF (Wind Axes)

Introduced in R2006a

3x3 Cross Product

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*.

$$A = a_1\mathbf{i} + a_2\mathbf{j} + a_3\mathbf{k}$$
$$B = b_1\mathbf{i} + b_2\mathbf{j} + b_3\mathbf{k}$$
$$C = A \times B = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix}$$
$$= (a_2b_3 - a_3b_2)\mathbf{i} + (a_3b_1 - a_1b_3)\mathbf{j} + (a_1b_2 - a_2b_1)\mathbf{k}$$

Inputs and Outputs

Input	Dimension Type	Description
First	3-by-1 vector	

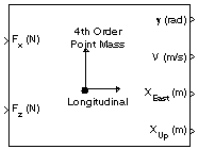
Input	Dimension Type	Description
Second	3-by-1 vector	

Output	Dimension Type	Description
First	3-by-1 vector	

Introduced before R2006a

4th Order Point Mass (Longitudinal)

Calculate fourth-order point mass

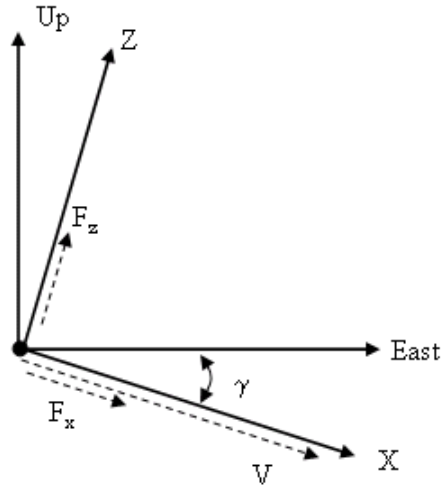


Library

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 $[X_{\text{East}}\ X_{\text{Up}}]^{\text{T}}$ are functions of airspeed (V) and flight path angle (γ),

$$\begin{aligned} F_x &= m\dot{V} \\ F_z &= mV\dot{\gamma} \\ \dot{X}_{East} &= V\cos\gamma \\ \dot{X}_{Up} &= V\sin\gamma \end{aligned}$$

where the applied forces $[F_x\ F_z]^{\text{T}}$ 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.

Parameters

Units

Specifies the input and output units:

Units	Forces	Velocity	Position
`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

Input	Dimension Type	Description
First		Contains the force in *x*-axis in selected units.
Second		Contains the force in *z*-axis in selected units.

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)

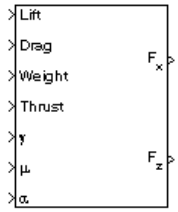
Simple Variable Mass 3DOF (Body Axes)

Simple Variable Mass 3DOF (Wind Axes)

Introduced before R2006a

4th Order Point Mass Forces (Longitudinal)

Calculate forces used by fourth-order point mass

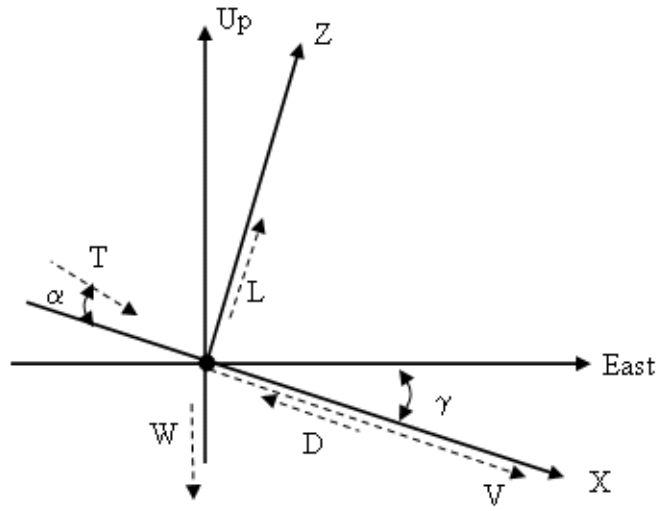


Library

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 (γ), angle of attack (α), and bank angle (μ).

$$F_z = (L + T\sin\alpha)\cos\mu - W\cos\gamma$$
$$F_x = T\cos\alpha - D - W\sin\gamma$$

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.

Input	Dimension Type	Description
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)

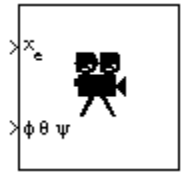
6th Order Point Mass (Coordinated Flight)

6th Order Point Mass Forces (Coordinated Flight)

Introduced before R2006a

6DoF Animation

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.

Parameters

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.

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

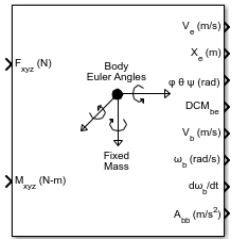
FlightGear Preconfigured 6DoF Animation

The figure axes properties CameraPosition and CameraViewAngle

Introduced before R2006a

6DOF (Euler Angles)

Implement Euler angle representation of six-degrees-of-freedom equations of motion

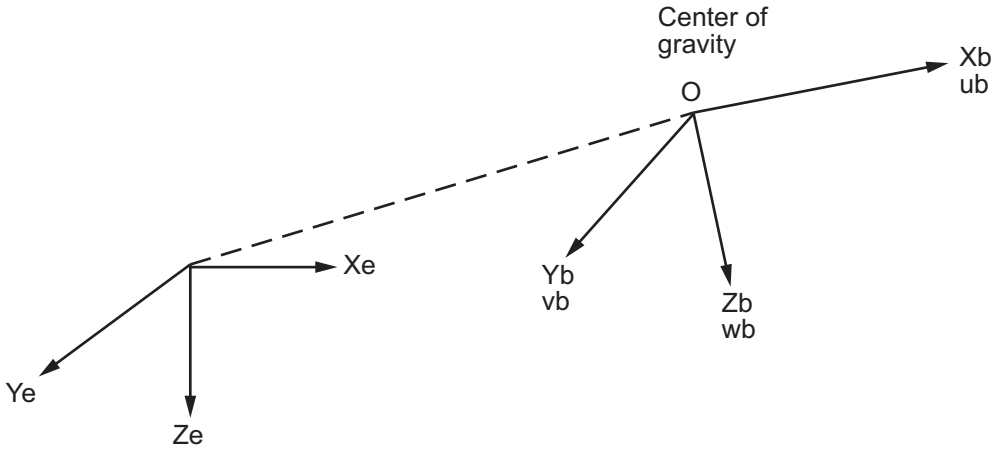


Library

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.

$$\bar{F}_b = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = m\left(\dot{\bar{V}}_b + \bar{\omega} \times \bar{V}_b\right)$$

$$A_{bb} = \begin{bmatrix} \dot{u}_b \\ \dot{v}_b \\ \dot{w}_b \end{bmatrix} = \frac{1}{m}\bar{F}_b - \bar{\omega} \times \bar{V}_b$$

$$A_{be} = \frac{1}{m}F_b$$

$$\bar{V}_b = \begin{bmatrix} u_b \\ v_b \\ w_b \end{bmatrix}, \bar{\omega} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

A_{bb} is the acceleration of the body with respect to the body reference frame. A_{bi} is the acceleration of the body with respect to an inertial reference frame.

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.

$$\bar{M}_B = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = I\dot{\bar{\omega}} + \bar{\omega} \times (I\bar{\omega})$$

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix}$$

The relationship between the body-fixed angular velocity vector, $[p\ q\ r]^T$, and the rate of change of the Euler angles, $\begin{bmatrix} \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T$, can be determined by resolving the Euler rates into the body-fixed coordinate frame.

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \equiv J^{-1} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

Inverting J then gives the required relationship to determine the Euler rate vector.

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = J \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & (\sin\phi\tan\theta) & (\cos\phi\tan\theta) \\ 0 & \cos\phi & -\sin\phi \\ 0 & \dfrac{\sin\phi}{\cos\theta} & \dfrac{\cos\phi}{\cos\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

Parameters

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.

Representation

Select the representation to use:

`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 mass of the rigid body.

Inertia

The 3-by-3 inertia tensor matrix I.

Include inertial acceleration

Select this check box to enable an additional output port for the accelerations in body-fixed axes with respect to the inertial frame. You typically connect this signal to the accelerometer.

Inputs and Outputs

Input	Dimension Type	Description
First	Vector	Contains the three applied forces in body-fixed coordinate frame.
Second	Vector	Contains the three applied moments in body-fixed coordinate frame.

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], within ±pi, 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.

Output	Dimension Type	Description
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 with respect to body frame.
Ninth (Optional)	Three-element vector	Contains the accelerations in body-fixed axes with respect to inertial frame (flat Earth). You typically connect this signal to the accelerometer.

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 `asbhl20` for examples of this block.

Reference

Stevens, Brian, and Frank Lewis, *Aircraft Control and Simulation*, Second Edition, John Wiley & Sons, 2003.

Zipfel, Peter H., *Modeling and Simulation of Aerospace Vehicle Dynamics*. Second Edition, AIAA Education Series, 2007.

See Also

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)

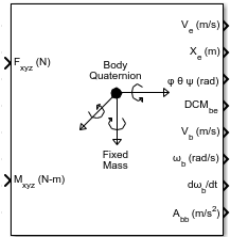
Simple Variable Mass 6DOF Wind (Quaternion)

Simple Variable Mass 6DOF Wind (Wind Angles)

Introduced in R2006a

6DOF (Quaternion)

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 ε 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.

$$\begin{bmatrix}\dot{q}_0\\ \dot{q}_1\\ \dot{q}_2\\ \dot{q}_3\end{bmatrix} = \frac{1}{2}\begin{bmatrix}0 & -p & -q & -r\\ p & 0 & r & -q\\ q & -r & 0 & p\\ r & q & -p & 0\end{bmatrix}\begin{bmatrix}q_0\\ q_1\\ q_2\\ q_3\end{bmatrix} + K\varepsilon\begin{bmatrix}q_0\\ q_1\\ q_2\\ q_3\end{bmatrix}$$

$$\varepsilon = 1 - \left(q_0^2 + q_1^2 + q_2^2 + q_3^2\right)$$

Parameters

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.

Representation

Select the representation to use:

`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 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.

Include inertial acceleration

Select this check box to enable an additional output port for the accelerations in body-fixed axes with respect to the inertial frame. You typically connect this signal to the accelerometer.

Inputs and Outputs

Input	Dimension Type	Description
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.

Output	Dimension Type	Description
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 with respect to body frame.
Ninth (Optional)	Three-element vector	Contains the accelerations in body-fixed axes with respect to inertial frame (flat Earth). You typically connect this signal to the accelerometer.

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, Brian, and Frank Lewis, *Aircraft Control and Simulation*, Second Edition, John Wiley & Sons, 2003.

Zipfel, Peter H., *Modeling and Simulation of Aerospace Vehicle Dynamics*. Second Edition, AIAA Education Series, 2007.

See Also

6DOF (Euler Angles)

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)

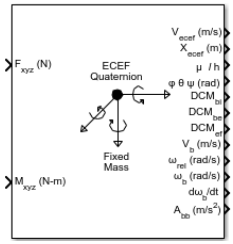
Simple Variable Mass 6DOF Wind (Quaternion)

Simple Variable Mass 6DOF Wind (Wind Angles)

Introduced in R2006a

6DOF ECEF (Quaternion)

Implement quaternion representation of six-degrees-of-freedom equations of motion in Earth-centered Earth-fixed (ECEF) coordinates

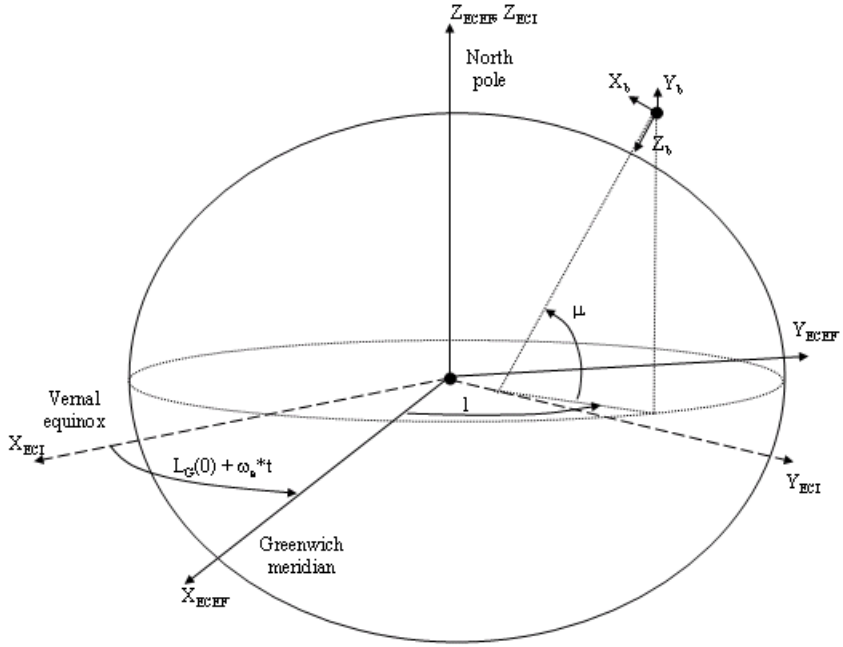


Library

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 (ω_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.



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.

$$\bar{F}_b = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = m\left(\dot{\bar{V}}_b + \bar{\omega}_b \times \bar{V}_b + DCM_{bf}\bar{\omega}_e \times \bar{V}_b + DCM_{bf}\left(\bar{\omega}_e \times \left(\bar{\omega}_e \times \bar{X}_f\right)\right)\right)$$

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.

$$A_{bb} = \begin{bmatrix} \dot{u}_b \\ \dot{v}_b \\ \dot{\omega}_b \end{bmatrix} = \frac{1}{m}\bar{F}_b - \left[\bar{\omega}_b \times \bar{V}_b + DCM_{bf}\bar{\omega}_e \times \bar{V}_b + DCM_{bf}\left(\bar{\omega}_e \times \left(\bar{\omega}_e \times \bar{X}_f\right)\right)\right]$$

$$A_{b\ ecef} = \frac{F_b}{m}$$

$$\bar{M}_b = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = I\dot{\bar{\omega}}_b + \bar{\omega}_b \times (I\bar{\omega}_b)$$

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix}$$

The integration of the rate of change of the quaternion vector is given below.

$$\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = -1/2 \begin{bmatrix} 0 & \omega_b(1) & \omega_b(2) & \omega_b(3) \\ -\omega_b(1) & 0 & -\omega_b(3) & \omega_b(2) \\ -\omega_b(2) & \omega_b(3) & 0 & -\omega_b(1) \\ -\omega_b(3) & -\omega_b(2) & \omega_b(1) & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

Parameters

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 (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, with latitude, longitude, and altitude. The altitude value depends on the selected units (meters (MKS) or feet (English)). Latitude and longitude values are in degrees and can be any value. However, latitude values of +90 and -90 may return unexpected values because of singularity at the poles.

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:

`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.

Include inertial acceleration

Select this check box to enable an additional output port for the accelerations in body-fixed axes with respect to the inertial frame. You typically connect this signal to the accelerometer.

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.
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.

Output	Dimension Type	Description
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 with respect to body frame.
Thirteenth (Optional)	Three-element vector	Contains the accelerations in body-fixed axes with respect to ECEF frame.

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 ±90 degrees, and longitude that lies between ±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)

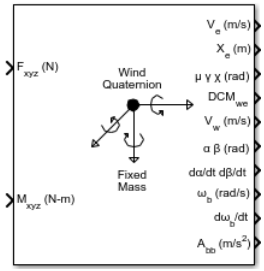
Simple Variable Mass 6DOF Wind (Quaternion)

Simple Variable Mass 6DOF Wind (Wind Angles)

Introduced in R2006a

6DOF Wind (Quaternion)

Implement quaternion representation of six-degrees-of-freedom equations of motion with respect to wind axes

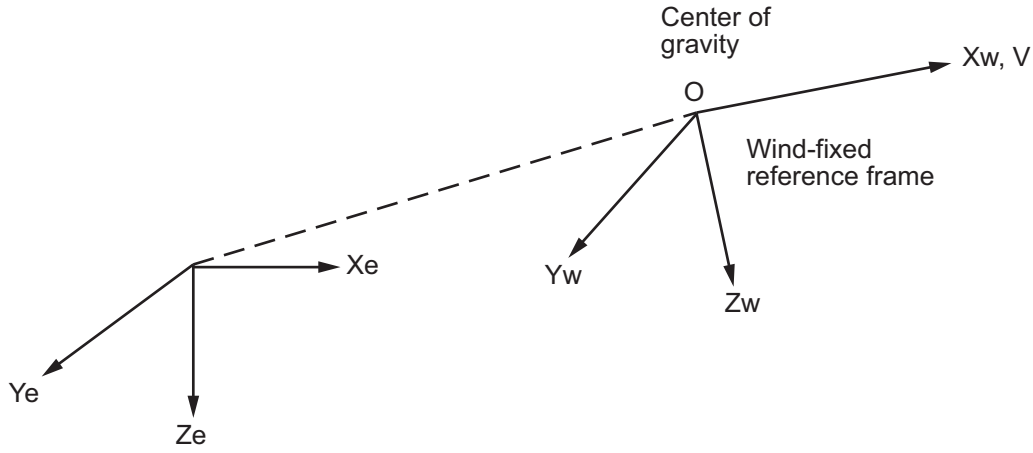


Library

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.



Flat Earth reference frame

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.

$$\bar{F}_w = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = m(\dot{\bar{V}}_w + \bar{\omega}_w \times \bar{V}_w)$$

$$A_{be} = DCM_{wb}\frac{\bar{F}_w}{m}$$

$$\bar{V}_w = \begin{bmatrix} V \\ 0 \\ 0 \end{bmatrix}, \bar{\omega}_w = \begin{bmatrix} p_w \\ q_w \\ r_w \end{bmatrix} = DMC_{wb}\begin{bmatrix} p_b - \dot{\beta}\sin\alpha \\ q_b - \dot{\alpha} \\ r_b + \dot{\beta}\cos\alpha \end{bmatrix}, \bar{\omega}_b = \begin{bmatrix} p_b \\ q_b \\ r_b \end{bmatrix}$$

$$A_{bb} = \begin{bmatrix} \dot{u}_b \\ \dot{v}_b \\ \dot{w}_b \end{bmatrix} = DCM_{wb}\left[\frac{\bar{F}_w}{m} - \bar{\omega}_w \times \bar{V}_w\right]$$

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.

$$\bar{M}_b = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = I\dot{\bar{\omega}}_b + \bar{\omega}_b \times (I\bar{\omega}_b)$$

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix}$$

The integration of the rate of change of the quaternion vector is given below.

$$\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = -1/2 \begin{bmatrix} 0 & p & q & r \\ -p & 0 & -r & q \\ -q & r & 0 & -p \\ -r & -q & p & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

Parameters

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.

Representation

Select the representation to use:

`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, 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.

Include inertial acceleration

Select this check box to enable an additional output port for the accelerations in body-fixed axes with respect to the inertial frame. You typically connect this signal to the accelerometer.

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.

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	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 with respect to body frame.

Output	Dimension Type	Description
Eleventh (Optional)	Three-element vector	Contains the accelerations in body-fixed axes with respect to inertial frame (flat Earth). You typically connect this signal to the accelerometer.

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 (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)

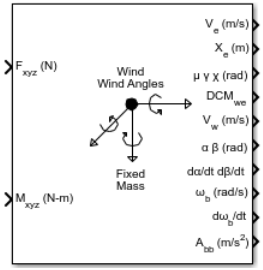
Simple Variable Mass 6DOF Wind (Quaternion)

Simple Variable Mass 6DOF Wind (Wind Angles)

Introduced in R2006a

6DOF Wind (Wind Angles)

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, $[\mu\ \gamma\chi]^{\mathrm{T}}$, 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.

$$\begin{bmatrix} \dot{\mu} \\ \dot{\gamma} \\ \dot{\chi} \end{bmatrix} = J \begin{bmatrix} p_w \\ q_w \\ r_w \end{bmatrix} = \begin{bmatrix} 1 & (\sin\mu\tan\gamma) & (\cos\mu\tan\gamma) \\ 0 & \cos\mu & -\sin\mu \\ 0 & \frac{\sin\mu}{\cos\gamma} & \frac{\cos\mu}{\cos\gamma} \end{bmatrix} DMC_{wb} \begin{bmatrix} p_b - \dot{\beta}\sin\alpha \\ q_b - \dot{\alpha} \\ r_b + \dot{\beta}\cos\alpha \end{bmatrix}$$

Parameters

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

Units	Forces	Moment	Acceleration	Velocity	Position	Mass	Inertia
`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.

Include inertial acceleration

Select this check box to enable an additional output port for the accelerations in body-fixed axes with respect to the inertial frame. You typically connect this signal to the accelerometer.

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.

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], within ±pi, 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.

Output	Dimension Type	Description
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 with respect to body frame.
Eleventh (Optional)	Three-element vector	Contains the accelerations in body-fixed axes with respect to inertial frame (flat Earth). You typically connect this signal to the accelerometer.

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)

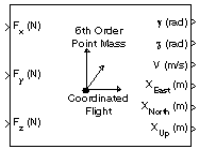
Simple Variable Mass 6DOF Wind (Quaternion)

Simple Variable Mass 6DOF Wind (Wind Angles)

Introduced in R2006a

6th Order Point Mass (Coordinated Flight)

Calculate sixth-order point mass in coordinated flight

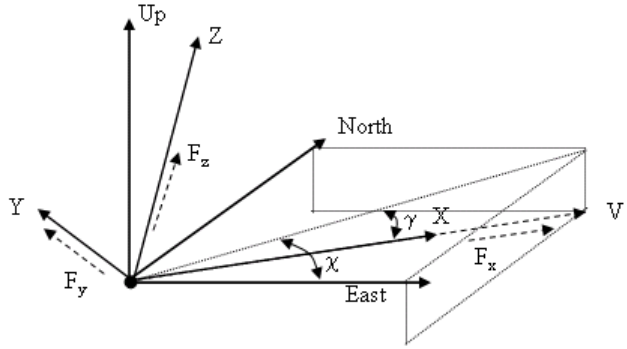


Library

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 (γ), and heading angle (χ),

$$F_x = mV$$
$$F_y = (mV\cos\gamma)\dot{\chi}$$
$$F_z = mV\dot{\gamma}$$
$$\dot{X}_{East} = V\cos\chi\cos\gamma$$
$$\dot{X}_{North} = V\sin\chi\cos\gamma$$
$$\dot{X}_{Up} = V\sin\gamma$$

where the applied forces $[F_x\ F_y\ F_h]^{\mathrm{T}}$ are in a system is defined by x-axis in the direction of vehicle velocity relative to air, z-axis is upward, and y-axis completes the right-handed frame, and the mass of the body m is assumed constant.

Parameters

Units

Specifies the input and output units:

Units	Forces	Velocity	Position
`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).

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)

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)

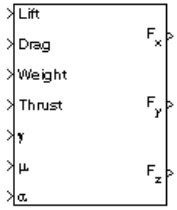
Simple Variable Mass 6DOF Wind (Quaternion)

Simple Variable Mass 6DOF Wind (Wind Angles)

Introduced before R2006a

6th Order Point Mass Forces (Coordinated Flight)

Calculate forces used by sixth-order point mass in coordinated flight

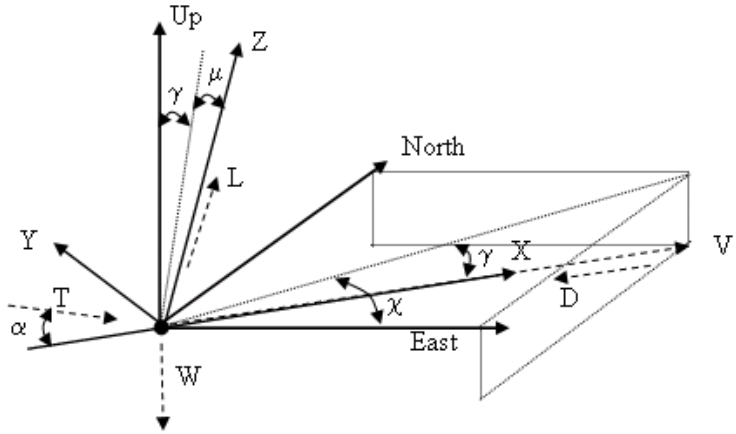


Library

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.



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 upwards and *y*-axis completes the right-handed frame and are functions of lift (L), drag (D), thrust (T), weight (W), flight path angle (γ), angle of attack (α), and bank angle (μ).

$$F_x = T\cos\alpha - D - W\sin\gamma$$
$$F\gamma = (L + T\sin\alpha)\sin\mu$$
$$F_z = (L + T\sin\alpha)\cos\mu - W\cos\gamma$$

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.

Input	Dimension Type	Description
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.

See Also

4th Order Point Mass (Longitudinal)

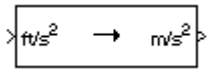
4th Order Point Mass Forces (Longitudinal)

6th Order Point Mass (Coordinated Flight)

Introduced before R2006a

Acceleration Conversion

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 unit** and **Final unit** lists.

Parameters

Initial unit

Specifies the input units.

Final unit

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^2	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

Introduced before R2006a

Adjoint of 3x3 Matrix

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

$$adj(A) = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix}$$

where

$$M_{ij} = (-1)^{i+j}$$

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

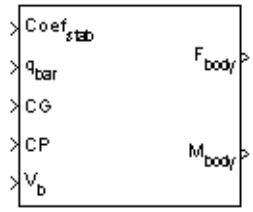
Determinant of 3x3 Matrix

Invert 3x3 Matrix

Introduced before R2006a

Aerodynamic Forces and Moments

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 α be the angle of attack and β 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.

$$\mathbf{F}_A^w \equiv \begin{bmatrix} -D \\ -C \\ -L \end{bmatrix} = C_{w\leftarrow b} \cdot \begin{bmatrix} X_A \\ Y_A \\ Z_A \end{bmatrix} \equiv C_{w\leftarrow b} \cdot \mathbf{F}_A^b$$

Components/Axes	x	y	z
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 ($C_N = -C_Z$).

Parameters

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)

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.

Input	Dimension Type	Description
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.

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

Introduced before R2006a

Airspeed Indicator

Display measurements for aircraft airspeed

Library

Flight Instruments

Description

The Airspeed Indicator block displays measurements for aircraft airspeed in knots.

By default, minor ticks represent 10-knot increments and major ticks represent 40-knot increments. The parameters **Minimum** and **Maximum** determine the minimum and maximum values on the gauge. The number and distribution of ticks is fixed, which means that the first and last tick display the minimum and maximum values. The ticks in between distribute evenly between the minimum and maximum values. For major ticks, the distribution of ticks is **Maximum**-**Minimum**/9. For minor ticks, the distribution of ticks is **Maximum**-**Minimum**/36.

The airspeed indicator has scale color bars that allow for overlapping for the first bar, displayed at a different radius. This different radius lets the block represent V_{FE} (maximum speed with flap extended) and V_{SO} (stall speed with flap extended) accurately for aircraft airspeed and stall speed.

Parameters

Connection

Select a 1-element signal to connect to the block.

To view the data from a signal, select a signal in the model. The signal appears in the **Connection** table. Select the option button next to the signal you want to display. Click **Apply** to connect the signal.

The table has a row for the signal connected to the block. If there are no signals selected in the model, or the block is not connected to any signals, the table is empty.

Minimum

Specify the minimum tick mark value.

Default value is 40.

Specify this number as a finite, real, double, scalar value.

The **Minimum** tick value must be less than the **Maximum** tick value.

Maximum

Specify the maximum tick mark value.

Default value is 400.

Specify this number as a finite, real, double, scalar value.

The **Minimum** tick value must be greater than the **Maximum** tick value.

Scale Colors

Specify ranges of color bands on the outside of the scale. Specify the minimum and maximum color range to display on the gauge.

Label

Name of connected signal.

- `Top`

  Show label at the top of the block.

- `Bottom`

  Show label at the bottom of the block.

- `Hide`

  Do not show the label or instructional text when the block is not connected.

See Also

Altimeter | Artificial Horizon | Climb Rate Indicator | Exhaust Gas Temperature (EGT) Indicator | Heading Indicator | Revolutions Per Minute (RPM) Indicator | Turn Coordinator

Topics

“Display Measurements with Cockpit Instruments” on page 2-47
“Flight Instrument Gauges” on page 2-46

Introduced in R2016a

Altimeter

Display measurements for aircraft altitude

Library

Flight Instruments

Description

The Altimeter Indicator block displays the altitude above sea level in feet, also known as the pressure altitude. The block displays the altitude value with needles on a gauge and a numeric indicator.

- The gauge has 10 major ticks. Within each major tick are five minor ticks. This gauge has three needles. Using the needles, the altimeter can display accurately only altitudes between 0 and 100,000 feet.
  - For the longest needle, an increment of a small tick represents 20 feet and a major tick represents 100 feet.
  - For the second longest needle, a minor tick represents 200 feet and a major tick represents 1,000 feet.
  - For the shortest needle a minor tick represents 2,000 feet and a major tick represents 10,000 feet.
- For the numeric display, the block shows values as numeric characters between 0 and 9,999 feet. When the numeric display value reaches 10,000 feet, the gauge displays the value as the remaining values below 10,000 feet. For example, 12,345 feet displays as 2,345 feet. When a value is less than 0 (below sea level), the block displays `0`. The

needles show the appropriate value except for when the value is below sea level or over 100000 feet. Below sea level, the needles set to `0`, over 100,000, the needles stay set at 100,000.

Parameters

Connection

Select a 1-element signal to connect to the block.

To view the data from a signal, select a signal in the model. The signal appears in the **Connection** table. Select the option button next to the signal you want to display. Click **Apply** to connect the signal.

The table has a row for the signal connected to the block. If there are no signals selected in the model, or the block is not connected to any signals, the table is empty.

Label

Name of connected signal.

- `Top`

  Show label at the top of the block.

- `Bottom`

  Show label at the bottom of the block.

- `Hide`

  Do not show the label or instructional text when the block is not connected.

See Also

Airspeed Indicator | Artificial Horizon | Climb Rate Indicator | Exhaust Gas Temperature (EGT) Indicator | Heading Indicator | Revolutions Per Minute (RPM) Indicator | Turn Coordinator

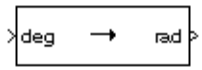
Topics

“Display Measurements with Cockpit Instruments” on page 2-47
“Flight Instrument Gauges” on page 2-46

Introduced in R2016a

Angle Conversion

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 unit** and the **Final unit** lists.

Parameters

Initial unit

Specifies the input units.

Final unit

Specifies the output units.

The following conversion units are available:

`deg`	Degrees
`rad`	Radians
`rev`	Revolutions

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

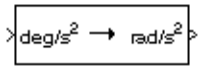
Temperature Conversion

Velocity Conversion

Introduced before R2006a

Angular Acceleration Conversion

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 unit** and the **Final unit** lists.

Parameters

Initial unit

Specifies the input units.

Final unit

Specifies the output units.

The following conversion units are available:

`deg/s`^{2}	Degrees per second squared
`rad/s`^{2}	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

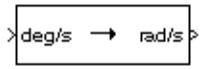
Temperature Conversion

Velocity Conversion

Introduced before R2006a

Angular Velocity Conversion

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 unit** and the **Final unit** lists.

Parameters

Initial unit

Specifies the input units.

Final unit

Specifies the output units.

The following conversion units are available:

`deg`/s	Degrees per second
`rad`/s	Radians per second
`rpm`	Revolutions per minute

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

Introduced before R2006a

Artificial Horizon

Represent aircraft attitude relative to horizon

Library

Flight Instruments

Description

The Artificial Horizon block represents aircraft attitude relative to horizon and displays roll and pitch in degrees:

- Values for roll cannot exceed +/– 90 degrees.
- Values for pitch cannot exceed +/– 30 degrees.

If the values exceed the maximum values, the gauge maximum and minimum values do not change.

Changes in roll value affect the gauge semicircles and the ticks located on the black arc turn accordingly. Changes in pitch value affect the scales and the distribution of the semicircles.

Combine the roll and pitch signals in a Mux block in the order:

1. Roll
2. Pitch

Parameters

Connection

Select a 2-element signal, consisting of signals for roll and pitch in degrees, to connect and display. This input cannot be a bus signal.

To view the data from a signal, select a signal in the model. The signal appears in the **Connection** table. Select the option button next to the signal you want to display. Click **Apply** to connect the signal.

The table has a row for the signal connected to the block. If there are no signals selected in the model, or the block is not connected to any signals, the table is empty.

Label

Name of connected signal.

- `Top`

  Show label at the top of the block.

- `Bottom`

  Show label at the bottom of the block.

- `Hide`

  Do not show the label or instructional text when the block is not connected.

See Also

Airspeed Indicator | Altimeter | Climb Rate Indicator | Exhaust Gas Temperature (EGT) Indicator | Heading Indicator | Revolutions Per Minute (RPM) Indicator | Turn Coordinator

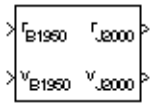
Topics

“Display Measurements with Cockpit Instruments” on page 2-47
“Flight Instrument Gauges” on page 2-46

Introduced in R2016a

Besselian Epoch to Julian Epoch

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 $(\bar{M}_{rr}, \bar{M}_{vr}, \bar{M}_{rv}, M_{vv})$ are defined as:

$$\bar{M}_{rr} \begin{bmatrix} 0.9999256782 & -0.0111820611 & -0.0048579477 \\ 0.0111820610 & 0.9999374784 & -0.0000271765 \\ 0.0048579479 & -0.0000271474 & 0.9999881997 \end{bmatrix}$$

$$\bar{M}_{vr} = \begin{bmatrix} 0.00000242395018 & -0.00000002710663 & -0.00000001177656 \\ 0.00000002710663 & 0.00000242397878 & -0.00000000006587 \\ 0.00000001177656 & -0.00000000006582 & 0.00000242410173 \end{bmatrix}$$

$$\bar{M}_{rv} = \begin{bmatrix} -0.000551 & -0.238565 & 0.435739 \\ 0.238514 & -0.002667 & -0.008541 \\ -0.435623 & 0.012254 & 0.002117 \end{bmatrix}$$

$$\bar{M}_{vv} = \begin{bmatrix} 0.99994704 & -0.01118251 & -0.00485767 \\ 0.01118251 & 0.99995883 & -0.00002718 \\ 0.00485767 & -0.00002714 & 1.00000956 \end{bmatrix}$$

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.

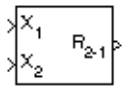
See Also

Julian Epoch to Besselian Epoch

Introduced before R2006a

Calculate Range

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

$$Range = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

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

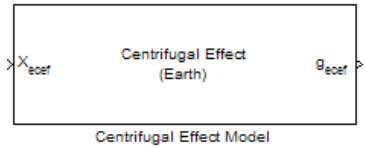
The calculated range is the magnitude of the distance, but not the direction. Therefore it is always positive or zero.

Craft positions are real values.

Introduced before R2006a

Centrifugal Effect Model

Implement mathematical representation of centrifugal effect for planetary gravity



Centrifugal Effect Model

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.

Parameters

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.

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. If **Planet model** has a value of `Earth`, this matrix contains Earth-centered Earth-fixed (ECEF) coordinates. The block does not use explicit units.

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 input distance units per second squared.

References

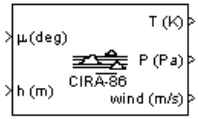
Vallado, D. A., *Fundamentals of Astrodynamics and Applications*, McGraw-Hill, New York, 1997.

NIMA TR8350.2: *Department of Defense World Geodetic System 1984, Its Definition and Relationship with Local Geodetic Systems*.

Introduced in R2010a

CIRA-86 Atmosphere Model

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.

Parameters

Units

Specifies the input and output units:

Units	Height	Temperature	Speed of Sound	Air Pressure	Air Density
`Metric (MKS)`	Meters	Kelvin	Meters per second	Pascal	Kilograms per cubic meter

Units	Height	Temperature	Speed of Sound	Air Pressure	Air Density
`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: • Geopotential heights in selected length units (**Coordinate type** is `GPHeight`) • Pressures in selected pressure units (**Coordinate type** is `Pressure`)

Output	Dimension Type	Description
First	Array	Contains mean temperature in selected units.
Second	Array	Contains an `m` array of either: • Pressures in selected pressure units (**Coordinate type** is `GPHeight`) • Geopotential heights in selected length units (**Coordinate type** is `Pressure`)
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 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., Shoeberl, 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://ccmc.gsfc.nasa.gov/modelweb/atmos/cospar1.html`

See Also

COESA Atmosphere Model

ISA Atmosphere Model

Introduced in R2007b

Climb Rate Indicator

Display measurements for aircraft climb rate

Library

Flight Instruments

Description

The Climb Rate Indicator block displays measurements for an aircraft climb rate in ft/min.

The needle covers the top semicircle, if the velocity is positive, and the lower semicircle, if the climb rate is negative. The range of the indicator is from –**Maximum** feet per minute to **Maximum** feet per minute. Major ticks indicate **Maximum**/4. Minor ticks indicate **Maximum**/8 and **Maximum**/80.

Parameters

Connection

Select a 1-element signal to connect to the block.

To view the data from a signal, select a signal in the model. The signal appears in the **Connection** table. Select the option button next to the signal you want to display. Click **Apply** to connect the signal.

The table has a row for the signal connected to the block. If there are no signals selected in the model, or the block is not connected to any signals, the table is empty.

Maximum

Specify the maximum tick mark value.

Default value is 4,000 ft/min.

Specify this number as a finite, real, double, scalar value.

The **Minimum** tick value is always `0`.

Label

Name of connected signal.

- `Top`

  Show label at the top of the block.
- `Bottom`

  Show label at the bottom of the block.
- `Hide`

  Do not show the label or instructional text when the block is not connected.

See Also

Airspeed Indicator | Altimeter | Artificial Horizon | Exhaust Gas Temperature (EGT) Indicator | Heading Indicator | Revolutions Per Minute (RPM) Indicator | Turn Coordinator

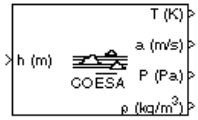
Topics

“Display Measurements with Cockpit Instruments” on page 2-47
“Flight Instrument Gauges” on page 2-46

Introduced in R2016a

COESA Atmosphere Model

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.

Parameters

Units

Specifies the input and output units:

Units	Height	Temperature	Speed of Sound	Air Pressure	Air Density
`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. Selecting `MIL-HDBK-310` enables the parameters **Atmospheric model type**, **Extreme parameter**, **Frequency of occurrence**, and **Altitude of extreme value**.
`MIL-STD-210C`	This selection is linked to the Non-Standard Day 210C block. See the block reference for more information. Selecting `MIL-HDBK-310` enables the parameters **Atmospheric model type**, **Extreme parameter**, **Frequency of occurrence**, and **Altitude of extreme value**.

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.

Selecting **Specification > MIL-HDBK-310** or **Specification > MIL-STD-210C** enables this parameter.

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 **Atmospheric model type**
`Low pressure`	This option is available only when `Envelope` is selected for **Atmospheric model type**

Selecting **Specification > MIL-HDBK-310** or **Specification > MIL-STD-210C** enables this parameter.

Frequency of occurrence

Select percent of time the values would occur.

`Extreme values`	This option is available only when `Envelope` is selected for **Atmospheric model type**.
`1%`	Option always available
`5%`	This option is available only when `Envelope` is selected for **Atmospheric model type**.
`10%`	Option always available
`20%`	This option is available only when `Envelope` is selected for **Atmospheric model type**.

Selecting **Specification > MIL-HDBK-310** or **Specification > MIL-STD-210C** enables this parameter.

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)`

Selecting **Specification** > **MIL-HDBK-310** or **Specification** > **MIL-STD-210C** enables this parameter.

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

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

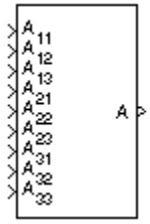
Non-Standard Day 210C

Non-Standard Day 310

Introduced before R2006a

Create 3x3 Matrix

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. This block accepts all data types that Simulink supports.

The output matrix has the form of

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}$$

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.

Input	Dimension Type	Description
Third		Contains the first row and third column of the matrix.
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

Introduced before R2006a

Crossover Pilot Model

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 on page 4-154]). 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.
ω_c	Crossover frequency.

Variable	Description
τ	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.

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(gcb,'pade','3')
```

Parameters

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.

Option (Controlled Element Transfer Function)	Transfer Function of Controlled Element (Y_c)	Transfer Function of Pilot (Y_p)	Y_cY_p	Notes
`Proportional`	K_c	$\frac{K_p e^{-\tau s}}{s}$	$\frac{K_c K_p e^{-\tau s}}{s}$	

Option (Controlled Element Transfer Function)	Transfer Function of Controlled Element (Y_c)	Transfer Function of Pilot (Y_p)	Y_cY_p	Notes
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 $\omega_n > 1/\tau$
Accelera-tion (*)	$\frac{K_c}{s^2}$	$K_p s e^{-\tau s}$	$\frac{K_c K_p e^{-\tau s}}{s}$	
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 $T_L \approx T_I$
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 $\omega_n \ll 1/\tau$, $1/T_L \approx \zeta\omega_n$

* 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.
τ	Pilot time delay.
T_I	Lag constant.
T_L	Lead constant.
ζ	Damping ratio for the aircraft.
ω_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.

Pilot lag constant

Specifies the pilot lag constant.

Pilot lead constant

Specifies the pilot lead constant.

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 188, Jan. 1974.

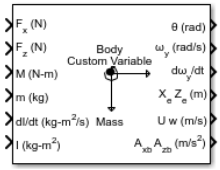
See Also

Precision Pilot Model | Tustin Pilot Model

Introduced in R2012b

Custom Variable Mass 3DOF (Body Axes)

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

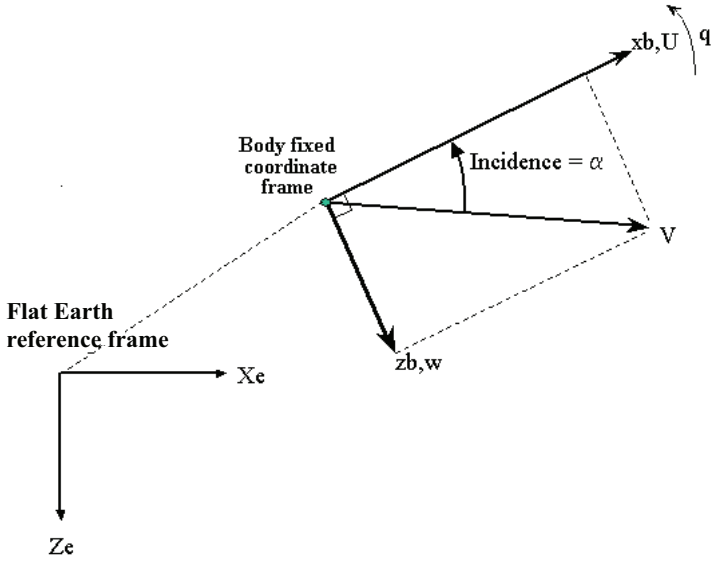


Library

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 a flat Earth reference frame.



The equations of motion are

$$A_{xb} = \dot{u} = \frac{F_x}{m} - \frac{\dot{m} U r e_b}{m} - qw - g \sin\theta$$

$$V r e_b = \begin{bmatrix} U r e & W r e \end{bmatrix}_b$$

$$A_{yb} = \dot{w} = \frac{F_z}{m} - \frac{\dot{m} W r e_b}{m} + qu + g \cos\theta$$

$$\dot{q} = \frac{M - \dot{I}_{yy} q}{I_{yy}}$$

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

$$A_{be} = \begin{bmatrix} A_{xe} \\ A_{ze} \end{bmatrix} = \frac{F_b - \dot{m} U r e_b}{m} - \bar{g}$$

$$A_{bb} = \begin{bmatrix} A_{xb} \\ A_{yb} \end{bmatrix}$$

where the applied forces are assumed to act at the center of gravity of the body. Ure_b and Wre_b are the relative velocities of the mass flow ($\dot{m}$) being added to or ejected from the body in body-fixed axes.

Parameters

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 `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, (θ_0).

Initial incidence

A scalar value for the initial angle between the velocity vector and the body, (α_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 flat Earth reference frame.

Gravity Source

Specify source of gravity:

`External`	Variable gravity input to block
`Internal`	Constant gravity specified in **Acceleration due to gravity**

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`. This parameter appears if you set **Gravity source** to `Internal`.

Include mass flow relative velocity

Select this check box to add a mass flow relative velocity port. This is the relative velocity at which the mass is accreted or ablated.

Include inertial acceleration

Select this check box to enable an additional output port for the accelerations in body-fixed axes with respect to the inertial frame. You typically connect this signal to the accelerometer.

Inputs and Outputs

Input	Dimension Type	Description
First	Scalar	Contains the force acting along the body x-axis, (F_x).
Second	Scalar	Contains the force acting along the body z-axis, (F_z).

Input	Dimension Type	Description
Third	Scalar	Contains the applied pitch moment, (M).
Fourth (Optional)	Vector	Contains the rate of change of mass, ($\dot{m}$) (positive if accreted, negative if ablated).
Fifth	Scalar	Contains the mass, (m).
Sixth	Scalar	Contains the rate of change of inertia tensor matrix, ($\dot{I}_{yy}$).
Seventh	Scalar	Contains the inertia tensor matrix, (I_{yy}).
Eighth (Optional)	Scalar	Contains the gravity in the selected units.
Ninth (Optional)	Two-element vector	Contains one or more relative velocities at which the mass is accreted to or ablated from the body in body axes.

Output	Dimension Type	Description
First	Scalar	Contains the pitch attitude, within ±pi, in radians (θ).
Second	Scalar	Contains the pitch angular rate, in radians per second (q).
Third	Scalar	Contains the pitch angular acceleration, in radians per second squared ($\dot{q}$).
Fourth	Two-element vector	Contains the location of the body, in the flat Earth reference frame, (Xe, Ze).
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, (Ax, Az).
Seventh (Optional)	Two-element vector	Contains the accelerations in body-fixed axes with respect to inertial frame (flat Earth). You typically connect this signal to the accelerometer.

See Also

3DOF (Body Axes)

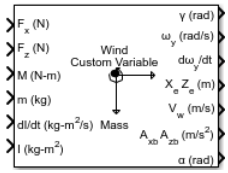
Incidence & Airspeed

Simple Variable Mass 3DOF (Body Axes)

Introduced in R2006a

Custom Variable Mass 3DOF (Wind Axes)

Implement three-degrees-of-freedom equations of motion of custom variable mass with respect to wind axes

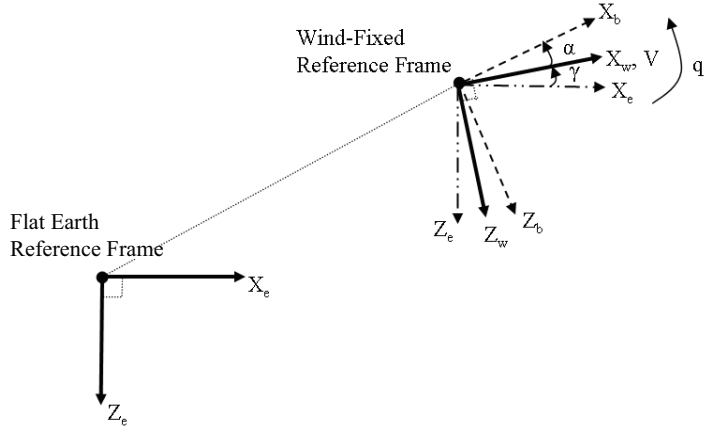


Library

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 a flat Earth reference frame.



The equations of motion are

$$\dot{V} = \frac{F_{x_{wind}}}{m} - \frac{\dot{m} V r e_{x_{wind}}}{m} - g \sin\gamma$$

$$A_{be} = \begin{bmatrix} A_{x_c} \\ A_{z_c} \end{bmatrix} = DCM_{wb} \left[\frac{F_w - \dot{m} V_{rew}}{m} - \bar{g} \right]$$

$$A_{bb} = \begin{bmatrix} A_{xb} \\ A_{zb} \end{bmatrix} = DCM_{wb} \left[\frac{F_w - \dot{m} V_{rew}}{m} - g - \bar{\omega}_w \times \bar{V}_w \right]$$

$$\dot{\alpha} = \frac{F_{z_{wind}}}{mV} + q + \frac{g}{V}\cos\gamma - \frac{\dot{m} V r e_{z_{wind}}}{mV}$$

$$\dot{q} = \dot{\theta} = \frac{M_{y_{body}} - \dot{I}_{yy} q}{I_{yy}}$$

$$\dot{\gamma} = q - \dot{\alpha}$$

where the applied forces are assumed to act at the center of gravity of the body. Vre_w is the relative velocity in the wind axes at which the mass flow ($\dot{m}$) is ejected or added to the body in wind axes.

Parameters

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

Units	Forces	Moment	Acceleration	Velocity	Position	Mass	Inertia
`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, (γ_0).

Initial incidence

A scalar value for the initial angle between the velocity vector and the body, (α_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 flat Earth reference frame.

Gravity Source

Specify source of gravity:

`External`	Variable gravity input to block
`Internal`	Constant gravity specified in **Acceleration due to gravity**

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`. This parameter appears if you set **Gravity source** to `Internal`.

Include mass flow relative velocity

Select this check box to add a mass flow relative velocity port. This is the relative velocity at which the mass is accreted or ablated.

Include inertial acceleration

Select this check box to enable an additional output port for the accelerations in body-fixed axes with respect to the inertial frame. You typically connect this signal to the accelerometer.

Inputs and Outputs

Input	Dimension Type	Description
First	Vector	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 one or more rates of change of mass, $(\dot{m})$ (positive if accreted, negative if ablated).
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.
Ninth (Optional)	Two-element vector	Contains one or more relative velocities at which the mass is accreted to or ablated from the body in wind axes.

Output	Dimension Type	Description
First		Contains the flight path angle, within ±pi, in radians (γ).

Output	Dimension Type	Description
Second		Contains the pitch angular rate, in radians per second (ω_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 flat Earth reference frame, (*Xe*, *Ze*).
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, (*Ax*, *Az*).
Seventh	Scalar	Contains the angle of attack, (α).
Eight (Optional)	Two-element vector	Contains the accelerations in body-fixed axes with respect to inertial frame (flat Earth). You typically connect this signal to the accelerometer.

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 (Body Axes)

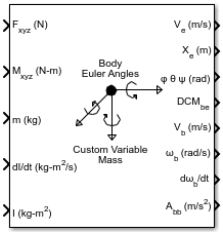
Simple Variable Mass 3DOF (Body Axes)

Simple Variable Mass 3DOF (Wind Axes)

Introduced in R2006a

Custom Variable Mass 6DOF (Euler Angles)

Implement Euler angle representation of six-degrees-of-freedom equations of motion of custom variable mass

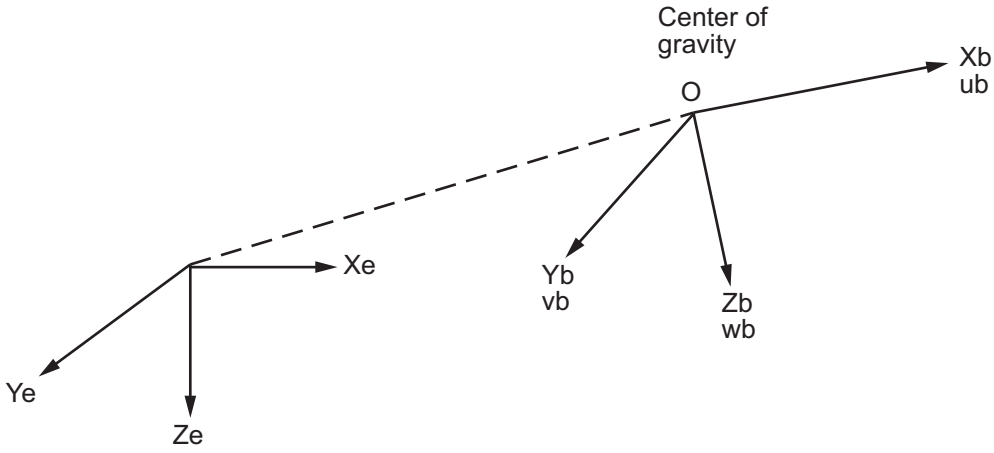


Library

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 $[F_x\ F_y\ F_z]^{\mathrm{T}}$ are in the body-fixed frame. Vre_{b} is the relative velocity in the body axes at which the mass flow ($\dot{m}$) is ejected or added to the body-fixed axes.

$$\bar{F}_b = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = m(\dot{\bar{V}}_b + \bar{\omega} \times \bar{V}_b) + \dot{m}\bar{V}re_b$$

$$A_{be} = \frac{\bar{F}_b - \dot{m}\bar{V}_{re_b}}{m}$$

$$A_{bb} = \begin{bmatrix} \dot{u}_b \\ \dot{v}_b \\ \dot{w}_b \end{bmatrix} = \frac{\bar{F}_b - \dot{m}\bar{V}_{re_b}}{m} - \bar{\omega} \times \bar{V}_b$$

$$\bar{V}_b = \begin{bmatrix} u_b \\ v_b \\ w_b \end{bmatrix}, \bar{\omega} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

The rotational dynamics of the body-fixed frame are given below, where the applied moments are $[L\ M\ N]^{\mathrm{T}}$, and the inertia tensor I is with respect to the origin O.

$$\bar{M}_B = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = I\dot{\bar{\omega}} + \bar{\omega} \times (I\bar{\omega}) + \dot{I}\bar{\omega}$$

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix}$$

$$\dot{I} = \begin{bmatrix} \dot{I}_{xx} & -\dot{I}_{xy} & -\dot{I}_{xz} \\ -\dot{I}_{yx} & \dot{I}_{yy} & -\dot{I}_{yz} \\ -\dot{I}_{zx} & -\dot{I}_{zy} & \dot{I}_{zz} \end{bmatrix}$$

The relationship between the body-fixed angular velocity vector, $[p\ q\ r]^{\mathrm{T}}$, and the rate of change of the Euler angles, $[\dot{\phi}\ \dot{\theta}\ \dot{\psi}]^{\mathrm{T}}$, can be determined by resolving the Euler rates into the body-fixed coordinate frame.

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} = J^{-1} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

Inverting J then gives the required relationship to determine the Euler rate vector.

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = J \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & (\sin\phi\tan\theta) & (\cos\phi\tan\theta) \\ 0 & \cos\phi & -\sin\phi \\ 0 & \dfrac{\sin\phi}{\cos\theta} & \dfrac{\cos\phi}{\cos\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

Parameters

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 `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.

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.

Include mass flow relative velocity

Select this check box to add a mass flow relative velocity port. This is the relative velocity at which the mass is accreted or ablated.

Include inertial acceleration

Select this check box to enable an additional output port for the accelerations in body-fixed axes with respect to the inertial frame. You typically connect this signal to the accelerometer.

Inputs and Outputs

Input	Dimension Type	Description
First	Vector	Contains the three applied forces.
Second	Vector	Contains the three applied moments.
Third (Optional)	Vector	Contains one or more rates of change of mass (positive if accreted, negative if ablated).
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.
Seventh (Optional)	Three-element vector	Contains one or more relative velocities at which the mass is accreted to or ablated from the body in body-fixed axes.

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], within ±pi, 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 with respect to body frame.
Ninth (Optional)	Three-element vector	Contains the accelerations in body-fixed axes with respect to inertial frame (flat Earth). You typically connect this signal to the accelerometer.

Assumptions and Limitations

The block assumes that the applied forces are acting at the center of gravity of the body.

Reference

Stevens, Brian, and Frank Lewis, *Aircraft Control and Simulation*, Second Edition, John Wiley & Sons, 2003.

Zipfel, Peter H., *Modeling and Simulation of Aerospace Vehicle Dynamics*. Second Edition, AIAA Education Series, 2007.

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 (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)

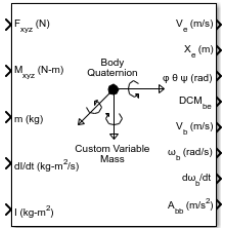
Simple Variable Mass 6DOF Wind (Quaternion)

Simple Variable Mass 6DOF Wind (Wind Angles)

Introduced in R2006a

Custom Variable Mass 6DOF (Quaternion)

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 ε 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.

$$\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} + K\varepsilon \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

$$\varepsilon = 1 - (q_0^2 + q_1^2 + q_2^2 + q_3^2).$$

Parameters

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 `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.

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.

Include mass flow relative velocity

Select this check box to add a mass flow relative velocity port. This is the relative velocity at which the mass is accreted or ablated.

Include inertial acceleration

Select this check box to enable an additional output port for the accelerations in body-fixed axes with respect to the inertial frame. You typically connect this signal to the accelerometer.

Inputs and Outputs

Input	Dimension Type	Description
First	Vector	Contains the three applied forces.
Second	Vector	Contains the three applied moments.
Third (Optional)	Vector	Contains one or more rates of change of mass (positive if accreted, negative if ablated).
Fourth	Scalar	Contains the mass.
Fifth	3-by-3 matrix	Contains rate of change of inertia tensor matrix.
Sixth	3-by-3 matrix	Contains the inertia tensor matrix.

Input	Dimension Type	Description
Seventh (Optional)	Three-element vector	Contains one or more relative velocities at which the mass is accreted to or ablated from the body in body-fixed axes.

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.
Eighth	Three-element vector	Contains the accelerations in body-fixed axes with respect to body frame.
Ninth (Optional)	Three-element vector	Contains the accelerations in body-fixed axes with respect to inertial frame (flat Earth). You typically connect this signal to the accelerometer.

Assumptions and Limitations

The block assumes that the applied forces are acting at the center of gravity of the body.

Reference

Stevens, Brian, and Frank Lewis, *Aircraft Control and Simulation*, Second Edition, John Wiley & Sons, 2003.

Zipfel, Peter H., *Modeling and Simulation of Aerospace Vehicle Dynamics*. Second Edition, AIAA Education Series, 2007.

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 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)

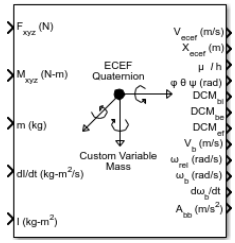
Simple Variable Mass 6DOF Wind (Quaternion)

Simple Variable Mass 6DOF Wind (Wind Angles)

Introduced in R2006a

Custom Variable Mass 6DOF ECEF (Quaternion)

Implement quaternion representation of six-degrees-of-freedom equations of motion of custom variable mass in Earth-centered Earth-fixed (ECEF) coordinates

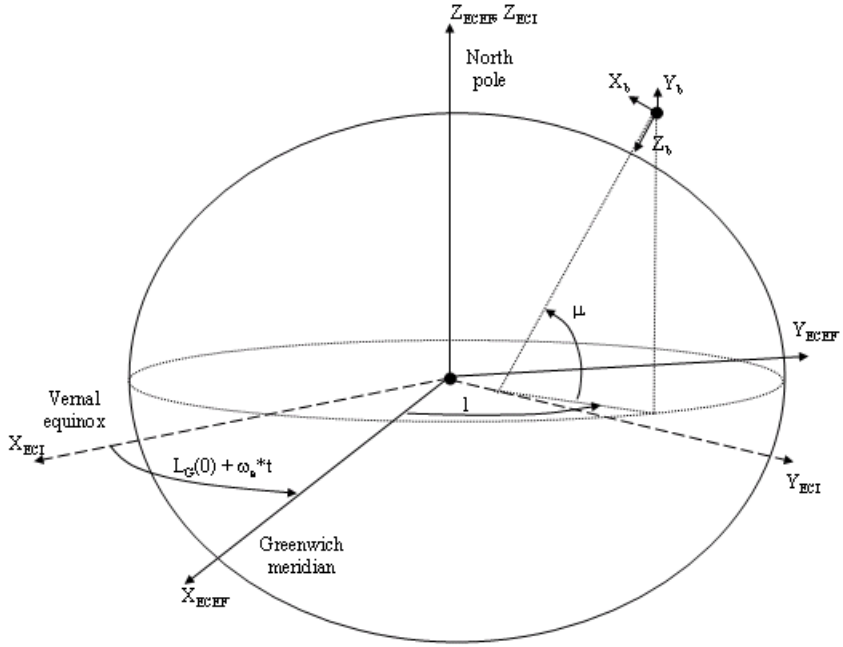


Library

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 (ω_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.



The translational motion of the ECEF coordinate frame is given below, where the applied forces $[F_x\ F_y\ F_z]^{\mathrm{T}}$ are in the body frame. Vre_{b} is the relative velocity in the wind axes at which the mass flow ($\dot{m}$) is ejected or added to the body in body-fixed axes.

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.

$$\bar{M}_b = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = \bar{I}\dot{\bar{\omega}}_b + \bar{\omega}_b \times (\bar{I}\bar{\omega}_b) + \dot{I}\bar{\omega}_b$$

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix}$$

The rate of change of the inertia tensor is defined by the following equation.

$$\dot{I} = \begin{bmatrix} \dot{I}_{xx} & -\dot{I}_{xy} & -\dot{I}_{xz} \\ -\dot{I}_{yx} & \dot{I}_{yy} & -\dot{I}_{yz} \\ -\dot{I}_{zx} & -\dot{I}_{zy} & \dot{I}_{zz} \end{bmatrix}$$

The integration of the rate of change of the quaternion vector is given below.

$$\begin{bmatrix}\dot{q}_0\\ \dot{q}_1\\ \dot{q}_2\\ \dot{q}_3\end{bmatrix} = -1/2\begin{bmatrix}0 & \omega_b(1) & \omega_b(2) & \omega_b(3)\\ -\omega_b(1) & 0 & -\omega_b(3) & \omega_b(2)\\ -\omega_b(2) & \omega_b(3) & 0 & -\omega_b(1)\\ -\omega_b(3) & -\omega_b(2) & \omega_b(1) & 0\end{bmatrix}\begin{bmatrix}q_0\\ q_1\\ q_2\\ q_3\end{bmatrix}$$

Parameters

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 (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. Latitude and longitude values can be any value. However, latitude values of +90 and -90 may return unexpected values because of singularity at the poles.

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.

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.

Include mass flow relative velocity

Select this check box to add a mass flow relative velocity port. This is the relative velocity at which the mass is accreted or ablated.

Planet model

Specifies the planet model to use, `Custom` or `Earth (WGS84)`.

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`.

Flattening

Specifies the flattening of the planet. 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.

Celestial longitude of Greenwich

The initial angle between Greenwich meridian and the *x*-axis of the ECI frame.

Include inertial acceleration

Select this check box to enable an additional output port for the accelerations in body-fixed axes with respect to the inertial frame. You typically connect this signal to the accelerometer.

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 (Optional)	Vector	Contains one or more rates of change of mass (positive if accreted, negative if ablated).
Fourth	Scalar	Contains the mass.
Sixth	3-by-3 matrix	Applies to the inertia tensor matrix.
Seventh (Optional)	1-by-1-by-*m* array	Contains one or more relative velocities at which the mass is accreted to or ablated from the body in body-fixed axes. *m* is three times the size of the third input vector.

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.

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 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 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 with respect to body frame.
Thirteenth (Optional)	Three-element vector	Contains the accelerations in body-fixed axes with respect to ECEF frame.

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 ±90 degrees, and longitude that lies between ±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)

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)

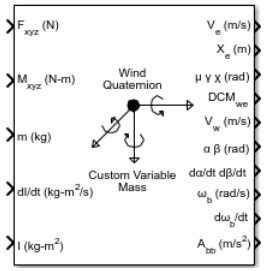
Simple Variable Mass 6DOF Wind (Quaternion)

Simple Variable Mass 6DOF Wind (Wind Angles)

Introduced in R2006a

Custom Variable Mass 6DOF Wind (Quaternion)

Implement quaternion representation of six-degrees-of-freedom equations of motion of custom variable mass with respect to wind axes

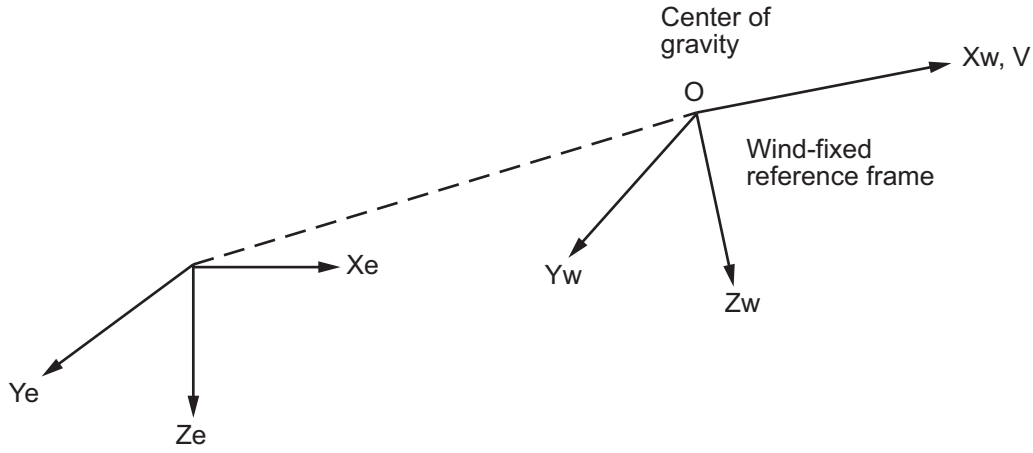


Library

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.



Flat Earth reference frame

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. Vre_w is the relative velocity in the wind axes at which the mass flow ($\dot{m}$) is ejected or added to the body.

$$\bar{F}_w = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = m(\dot{\bar{V}}_w + \bar{\omega}_w \times \bar{V}_w) + \dot{m}\bar{V}re_w$$

$$A_{be} = DCM_{wb}\frac{\left[\bar{F}_w - \dot{m}V_{re}\right]}{m}$$

$$\bar{V}_w = \begin{bmatrix} V \\ 0 \\ 0 \end{bmatrix}, \bar{\omega}_w = \begin{bmatrix} p_w \\ q_w \\ r_w \end{bmatrix} = DMC_{wb}\begin{bmatrix} p_b - \dot{\beta}\sin\alpha \\ q_b - \dot{\alpha} \\ r_b + \dot{\beta}\cos\alpha \end{bmatrix}, \bar{w}_b = \begin{bmatrix} p_b \\ q_b \\ r_b \end{bmatrix}$$

$$A_{bb} = DCM_{wb}\left[\frac{\bar{F}w - \dot{m}V_{re}}{m} - \bar{\omega}_w \times \bar{V}_w\right]$$

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.

$$\bar{M}_b = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = I\dot{\bar{\omega}}_b + \bar{\omega}_b \times (I\bar{\omega}_b) + \dot{I}\bar{\omega}_b$$

$$A_{bb} = \begin{bmatrix} \dot{U}_b \\ \dot{V}_b \\ \dot{W}_b \end{bmatrix} = DCM_{wb}\left[\frac{\overline{F}w - \dot{m}V_{re}}{m} - \overline{\omega}_w \times \overline{V}_w\right]$$

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix}$$

The integration of the rate of change of the quaternion vector is given below.

$$\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = -\frac{1}{2}\begin{bmatrix} 0 & p & q & r \\ -p & 0 & -r & q \\ -q & r & 0 & -p \\ -r & -q & p & 0 \end{bmatrix}\begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

Parameters

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

Units	Forces	Moment	Acceleration	Velocity	Position	Mass	Inertia
`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.

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.

Include mass flow relative velocity

Select this check box to add a mass flow relative velocity port. This is the relative velocity at which the mass is accreted or ablated.

Include inertial acceleration

Select this check box to enable an additional output port for the accelerations in body-fixed axes with respect to the inertial frame. You typically connect this signal to the accelerometer.

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 (Optional)	Vector	Contains one or more rates of change of mass, positive if accreted, negative if ablated.
Fourth	Scalar	Contains the mass of the body
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.
Seventh (Optional)	Three-element vector	Contains one or more relative velocities at which the mass is accreted to or ablated from the body in wind axes.

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.

Output	Dimension Type	Description
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 with respect to body frame.
Eleventh (Optional)	Three-element vector	Contains the accelerations in body-fixed axes with respect to inertial frame (flat Earth). You typically connect this signal to the accelerometer.

Assumptions and Limitations

The block assumes that the applied forces are acting at the center of gravity of the body.

References

Stevens, Brian, and Frank Lewis, *Aircraft Control and Simulation*, Second Edition, John Wiley & Sons, 2003.

Zipfel, Peter H., *Modeling and Simulation of Aerospace Vehicle Dynamics*. Second Edition, AIAA Education Series, 2007.

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)

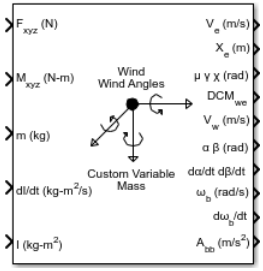
Simple Variable Mass 6DOF Wind (Quaternion)

Simple Variable Mass 6DOF Wind (Wind Angles)

Introduced in R2006a

Custom Variable Mass 6DOF Wind (Wind Angles)

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.

$$\begin{bmatrix} \dot{\mu} \\ \dot{\gamma} \\ \dot{\chi} \end{bmatrix} = J \begin{bmatrix} p_w \\ q_w \\ r_w \end{bmatrix} = \begin{bmatrix} 1 & (\sin\mu\tan\gamma) & (\cos\mu\tan\gamma) \\ 0 & \cos\mu & -\sin\mu \\ 0 & \frac{\sin\mu}{\cos\gamma} & \frac{\cos\mu}{\cos\gamma} \end{bmatrix} DMC_{wb} \begin{bmatrix} p_b - \dot{\beta}\sin\alpha \\ q_b - \dot{\alpha} \\ r_b + \dot{\beta}\cos\alpha \end{bmatrix}$$

Parameters

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

Units	Forces	Moment	Acceleration	Velocity	Position	Mass	Inertia
`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.

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.

Include mass flow relative velocity

Select this check box to add a mass flow relative velocity port. This is the relative velocity at which the mass is accreted or ablated.

Include inertial acceleration

Select this check box to enable an additional output port for the accelerations in body-fixed axes with respect to the inertial frame. You typically connect this signal to the accelerometer.

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 (Optional)	Vector	Contains one or more rates of change of mass (positive if accreted, negative if ablated).
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.
Seventh (Optional)	1-by-1-by-*m* array	Contains one or more relative velocities at which the mass is accreted to or ablated from the body in wind axes. *m* is three times the size of the third input vector.

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], within ±pi, 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 with respect to body frame.
Eleventh (Optional)	Three-element vector	Contains the accelerations in body-fixed axes with respect to inertial frame (flat Earth). You typically connect this signal to the accelerometer.

Assumptions and Limitations

The block assumes that the applied forces are acting at the center of gravity of the body.

References

Stevens, Brian, and Frank Lewis, *Aircraft Control and Simulation*, Second Edition, John Wiley & Sons, 2003.

Zipfel, Peter H., *Modeling and Simulation of Aerospace Vehicle Dynamics*. Second Edition, AIAA Education Series, 2007.

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)

Introduced in R2006a

Delta UT1

Calculate difference between principal Universal Time (UT1) and Coordinated Universal Time (UTC) according to International Astronomical Union (IAU) 2000A reference system

Library: Utilities / Axes Transformations

Description

The Delta UT1 block calculates the difference between principal UT1 and UTC according to the IAU 2000A reference system and Earth orientation data. By default, this block uses a prepopulated list of International Astronomical Union (IAU) 2000A Earth orientation (IERS) data. This list contains measured and calculated (predicted) data supplied by the IERS. The IERS measures and calculates this data for a set of predetermined dates. For dates after those listed in the prepopulated list, Delta UT1 calculates the data using this equation, limiting the values to +/- .9s:

```
UT1-UTC=0.5309-0.00123(MJD-57808)-(UT2-UT1)
```

Ports

Input

UTC — UT1 for UTC

modified Julian date

UT1 for UTC, specified as a modified Julian date. Use the `mjuliandate` function to convert the UTC date to a modified Julian date.

Data Types: `double`

Output Arguments

`ΔUT1` — Difference between UT1 and UTC
double

Difference between UT1 and UTC.

Data Types: `double`

Parameters

`IERS data file` — Earth orientation data
`aeroiersdata.mat` (default) | MAT-file

Custom list of Earth orientation data, specified in a MAT-file.

`Action for out-of-range input` — Out-of-range action
Warning (default) | Error | None

Out-of-range action, specified as a string.

Action to take in case of out-of-range or predicted value dates, specified as a string:

- Warning — Displays warning and indicates that the dates were out-of-range or predicted values.
- Error — Displays error and indicates that the dates were out-of-range or predicted values.
- None — Does not display warning or error.

`IERS data URL` — Website or Earth orientation data file
`http://maia.usno.navy.mil/ser7/finals2000A.data` (default) | website address | file name

Website or Earth orientation data file containing the Earth orientation data according to the IAU 2000A, specified as a website address or file name.

`Destination folder` — Folder for IERS data file
Current Folder (default)

Folder for IERS data file, specified as a character array or string. Before running this function, create *foldername* with write permission.

To create the IERS data file in the destination folder, click the **Create** button.

See Also

`aeroReadIERSData`

Topics

“Calculate UT1 to UTC Values” on page 2-50

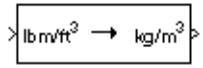
External Websites

http://maia.usno.navy.mil/ser7/finals2000A.data

Introduced in R2017b

Density Conversion

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 unit** and the **Final unit** lists.

Parameters

Initial unit

Specifies the input units.

Final unit

Specifies the output units.

The following conversion units are available:

`lbm/ft`^{3}	Pound mass per cubic foot
`kg/m`^{3}	Kilograms per cubic meter
`slug/ft`^{3}	Slugs per cubic foot
`lbm/in`^{3}	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

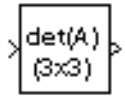
Temperature Conversion

Velocity Conversion

Introduced before R2006a

Determinant of 3x3 Matrix

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

$$det(A) = A_{11}(A_{22}A_{33} - A_{23}A_{32}) - A_{12}(A_{21}A_{33} - A_{23}A_{31}) + A_{13}(A_{21}A_{32} - A_{22}A_{31})$$

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

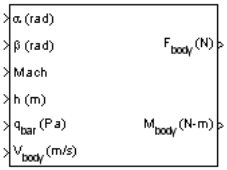
Create 3x3 Matrix

Invert 3x3 Matrix

Introduced before R2006a

Digital DATCOM Forces and Moments

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:

$$\boldsymbol{F} = \boldsymbol{F}_{\text{static}} + \boldsymbol{F}_{\text{dyn}}$$

$$\boldsymbol{M} = \boldsymbol{M}_{\text{static}} + \boldsymbol{M}_{\text{dyn}}$$

$\boldsymbol{F}_{\text{static}}$ and $\boldsymbol{M}_{\text{static}}$ are the static contribution, and $\boldsymbol{F}_{\text{dyn}}$ and $\boldsymbol{M}_{\text{dyn}}$ 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
$C_{n\beta}$	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.

$C_{D\ static} = C_D$

$C_{y\ static} = C_{Y\beta}\beta$

$C_{L\ static} = C_L$

$C_{l\ static} = C_{l\beta}\beta$

$C_{m\ static} = C_M$

$C_{n\ static} = C_{n\beta}\beta$

Dynamic Stability Characteristics

Dynamic stability characteristics include the following.

Coefficient	Meaning
C_{Lq}	Matrix of lift force derivatives due to pitch rate
C_{mq}	Matrix of pitching-moment derivatives due to pitch rate
$C_{Ld\alpha/dt}$	Matrix of lift force derivatives due to rate of angle of attack
$C_{md\alpha/dt}$	Matrix of pitching-moment derivatives due to rate of angle of attack

Coefficient	Meaning
C_{lp}	Matrix of rolling-moment derivatives due to roll rate
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.

$$C_{D\ dyn} = 0$$
$$C_{y\ dyn} = C_{yp} p (b_{ref} / 2V)$$
$$C_{L\ dyn} = (C_{Lq} q + C_{L\dot{\alpha}} \dot{\alpha})(c_{bar} / 2V)$$
$$C_{l\ dyn} = (C_{lp} p + C_{lr} r)(b_{ref} / 2V)$$
$$C_{m\ dyn} = (C_{mq} q + C_{m\dot{\alpha}} \dot{\alpha})(c_{bar} / 2V)$$
$$C_{n\ dyn} = (C_{np} p + C_{nr} r)(b_{ref} / 2V)$$

Parameters

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 structure

Specifies the MATLAB structure containing the digital DATCOM data. This structure is generated by the Aerospace Toolbox function `datcomimport`. To include dynamic derivatives in the generated output file, call the `datomimport` function with the damp keyword. The Digital DATCOM Forces and Moments block supports only Digital DATCOM, which is the 1976 version of DATCOM.

For more information on creating the digital DATCOM structure, see “Importing from USAF Digital DATCOM Files” (Aerospace Toolbox). This example shows how to bring United States Air Force (USAF) Digital DATCOM files into the MATLAB environment using the Aerospace Toolbox software.

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	Scalar	Contains the angle of attack.
Second	Scalar	Contains the sideslip angle, in radians.
Third	Scalar	Contains the Mach number.
Fourth	Scalar	Contains the altitude, in selected length units.
Fifth	Scalar	Contains the dynamic pressure, in selected pressure units.

Input	Dimension Type	Description
Sixth	Three-element vector	Contains the velocity, in selected velocity units and selected force axes.
Seventh (Optional)	Scalar	Contains the angle of attack rate, in radians per second. Appears when DAMP Control Card is used in input to Digital DATCOM.
Eight (Optional)	Three-element vector	Contains the body angular rates, in radians per second. Appears when DAMP Control Card is used in input to Digital DATCOM.
Ninth (Optional)	Scalar	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	Three-element vector	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	Three-element vector	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].

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.

Example

See "Importing from USAF Digital DATCOM Files" (Aerospace Toolbox) for an example of how to create and format DATCOM data for this block.

See `asbSkyHogg` for an example of this block.

See Also

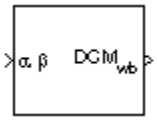
`datcomimport`

Aerodynamic Forces and Moments

Introduced in R2006b

Direction Cosine Matrix Body to Wind

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 (α) to axes (ox_1, oy_1, oz_1)
2. A rotation about oz_1 through the sideslip angle (β) 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.

$$DCM_{wb} = \begin{bmatrix} \cos\alpha\cos\beta & \sin\beta & \sin\alpha\cos\beta \\ -\cos\alpha\sin\beta & \cos\beta & -\sin\alpha\sin\beta \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix}$$

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

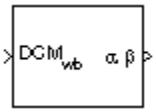
Rotation Angles to Direction Cosine Matrix

Wind Angles to Direction Cosine Matrix

Introduced before R2006a

Direction Cosine Matrix Body to Wind to Alpha and Beta

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 (α) to axes (ox_1, oy_1, oz_1)
2. A rotation about oz_1 through the sideslip angle (β) 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:

$$\alpha = \text{asin}(-DCM(3,1))$$

$$\beta = \text{asin}(DCM(1,2))$$

Parameters

Action for invalid DCM

Block behavior when direction cosine matrix is invalid (not orthogonal).

- Warning — Displays warning and indicates that the direction cosine matrix is invalid.
- Error — Displays error and indicates that the direction cosine matrix is invalid.
- None — Does not display warning or error (default).

Tolerance for DCM validation

Tolerance of direction cosine matrix validity, specified as a scalar. Default is `eps(2)`. The block considers the direction cosine matrix valid if these conditions are true:

- The transpose of the direction cosine matrix times itself equals 1 within the specified tolerance (`transpose(n)*n == 1±tolerance`)
- The determinant of the direction cosine matrix equals 1 within the specified tolerance (`det(n) == 1±tolerance`).

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 ±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 to Wind Angles

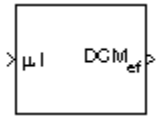
Rotation Angles to Direction Cosine Matrix

Wind Angles to Direction Cosine Matrix

Introduced before R2006a

Direction Cosine Matrix ECEF to NED

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 (ι) to axes (ox_1, oy_1, oz_1)
2. A rotation about oy_1 through the geodetic latitude (μ) to axes (ox_2, oy_2, oz_2)

$$\begin{bmatrix} ox_2 \\ oy_2 \\ oz_2 \end{bmatrix} = DCM_{ef} \begin{bmatrix} ox_0 \\ oy_0 \\ oz_0 \end{bmatrix}$$

$$\begin{bmatrix} ox_2 \\ oy_2 \\ oz_2 \end{bmatrix} = \begin{bmatrix} -\sin\mu & 0 & \cos\mu \\ 0 & 1 & 0 \\ -\cos\mu & 0 & -\sin\mu \end{bmatrix} \begin{bmatrix} \cos\iota & \sin\iota & 0 \\ -\sin\iota & \cos\iota & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} ox_0 \\ oy_0 \\ oz_0 \end{bmatrix}$$

Combining the two axis transformation matrices defines the following DCM.

$$DCM_{ef} = \begin{bmatrix} -\sin\mu\cos\iota & -\sin\mu\sin\iota & \cos\mu \\ -\sin\iota & \cos\iota & 0 \\ -\cos\mu\cos\iota & -\cos\mu\sin\iota & -\sin\mu \end{bmatrix}$$

Inputs and Outputs

Input	Dimension Type	Description
First	2-by-1 vector	Contains the geodetic latitude and longitude, in degrees. Latitude and longitude values can be any value. However, latitude values of +90 and -90 may return unexpected values because of singularity at the poles.

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

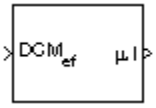
LLA to ECEF Position

Wind Angles to Direction Cosine Matrix

Introduced before R2006a

Direction Cosine Matrix ECEF to NED to Latitude and Longitude

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 (ι) to axes (ox_1, oy_1, oz_1)
2. A rotation about oy_1 through the geodetic latitude (μ) to axes (ox_2, oy_2, oz_2)

$$\begin{bmatrix} ox_2 \\ oy_2 \\ oz_2 \end{bmatrix} = DCM_{ef} \begin{bmatrix} ox_0 \\ oy_0 \\ oz_0 \end{bmatrix}$$

$$\begin{bmatrix} ox_2 \\ oy_2 \\ oz_2 \end{bmatrix} = \begin{bmatrix} -\sin\mu & 0 & \cos\mu \\ 0 & 1 & 0 \\ -\cos\mu & 0 & -\sin\mu \end{bmatrix} \begin{bmatrix} \cos\iota & \sin\iota & 0 \\ -\sin\iota & \cos\iota & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} ox_0 \\ oy_0 \\ oz_0 \end{bmatrix}$$

Combining the two axis transformation matrices defines the following DCM.

$$DCM_{ef} = \begin{bmatrix} -\sin\mu\cos\iota & -\sin\mu\sin\iota & \cos\mu \\ -\sin\iota & \cos\iota & 0 \\ -\cos\mu\cos\iota & -\cos\mu\sin\iota & -\sin\mu \end{bmatrix}$$

To determine geodetic latitude and longitude from the DCM, the following equations are used:

$$\mu = \text{asin}\left(-DCM(3,3)\right)$$

$$\iota = \text{atan}\left(\frac{-DCM(2,1)}{DCM(2,2)}\right)$$

Parameters

Action for invalid DCM

Block behavior when direction cosine matrix is invalid (not orthogonal).

- Warning — Displays warning and indicates that the direction cosine matrix is invalid.
- Error — Displays error and indicates that the direction cosine matrix is invalid.
- None — Does not display warning or error (default).

Tolerance for DCM validation

Tolerance of direction cosine matrix validity, specified as a scalar. Default is `eps(2)`. The block considers the direction cosine matrix valid if these conditions are true:

- The transpose of the direction cosine matrix times itself equals 1 within the specified tolerance (`transpose(n)*n == 1±tolerance`)
- The determinant of the direction cosine matrix equals 1 within the specified tolerance (`det(n) == 1±tolerance`).

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 ±90 degrees, and longitude that lies between ±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.

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

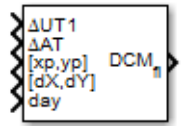
LLA to ECEF Position

Wind Angles to Direction Cosine Matrix

Introduced before R2006a

Direction Cosine Matrix ECI to ECEF

Convert Earth-centered inertial (ECI) to Earth-centered Earth-fixed (ECEF) coordinates



Library

Utilities/Axes Transformations

Description

The Direction Cosine Matrix ECI to ECEF block calculates the position direction cosine matrix (ECI to ECEF), based on the specified reduction method and Universal Coordinated Time (UTC), for the specified time and geophysical data.

Parameters

Reduction

Reduction method to calculate the direction cosine matrix. Method can be one of the following:

- `IAU-76/FK5`

  Reduce the calculation using the International Astronomical Union (IAU)-76/Fifth Fundamental Catalogue (FK5) (IAU-76/FK5) reference system. Choose this reduction method if the reference coordinate system for the conversion is FK5.

  Note This method uses the IAU 1976 precession model and the IAU 1980 theory of nutation to reduce the calculation. This model and theory are no longer current, but the software provides this reduction method for existing implementations.

Because of the polar motion approximation that this reduction method uses, the block calculates the transformation matrix rather than the direction cosine matrix.

- `IAU-2000/2006`

  Reduce the calculation using the International Astronomical Union (IAU)-2000/2006 reference system. Choose this reduction method if the reference coordinate system for the conversion is IAU-2000. This reduction method uses the P03 precession model to reduce the calculation.

Year

Specify the year used to calculate the Universal Coordinated Time (UTC) date. Enter a double value that is a whole number greater than 1, such as `2013`.

Month

Specify the month used to calculate the UTC date. From the list, select the month from `January` to `December`.

Day

Specify the day used to calculate the UTC date. From the list, select the day from `1` to `31`.

Hour

Specify the hour used to calculate the UTC date. Enter a double value that is a whole number, from `0` to `24`.

Minutes

Specify the minutes used to calculate the UTC date. Enter a double value that is a whole number, from `0` to `60`.

Seconds

Specify the seconds used to calculate the UTC date. Enter a double value that is a whole number, from `0` to `60`.

Time Increment

Specify the time increment between the specified date and the desired model simulation time. The block adjusts the calculated direction cosine matrix to take into account the time increment from model simulation. For example, selecting `Day` and connecting a simulation timer to the port means that each time increment unit is one day and the block adjusts its calculation based on that simulation time.

This parameter corresponds to the fifth block input, the clock source.

Possible values are `Day`, `Hour`, `Min`, `Sec`, and `None`. If you select `None`, the calculated Julian date does not take into account the model simulation time. Selecting this option removes the fifth block input.

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.

Higher accuracy parameters

Select this check box to enable the following inputs. These inputs let you better control the conversion result. See “Inputs and Outputs” on page 4-227 for a description.

$\Delta UT1$
ΔAT
$[xp,yp]$
$[\Delta\delta\psi, \Delta\delta\varepsilon]$ or $[dX,dY]$

Inputs and Outputs

Input	Dimension Type	Description
First (Optional)	Scalar	$\Delta UT1$, difference between UTC and Universal Time (UT1) in seconds, for which the function calculates the direction cosine or transformation matrix, for example, `0.234`.
Second (Optional)	Scalar	ΔAT, difference between International Atomic Time (IAT) and UTC in seconds, for which the function calculates the direction cosine or transformation matrix, for example, `32`.

Input	Dimension Type	Description
Third (Optional)	1-by-2 array	[*xp*,*yp*], polar displacement of the Earth, in radians, from the motion of the Earth crust, along the *x*- and *y*-axes, for example, `[-0.0682e-5 0.1616e-5]`
Fourth (Optional)	1-by-2 array	• If reduction method is `IAU-2000/2006`, this input is the adjustment to the location of the Celestial Intermediate Pole (CIP), specified in radians. This location ([d*X*,d*Y*]) is along the *x*- and *y*-axis, for example, `[-0.2530e-6 -0.0188e-6]`. • If reduction method is `IAU-76/FK5`, this input is the adjustment to the longitude ([Δδψ, Δδε]), specified in radians, for example, `[-0.2530e-6 -0.0188e-6]`. For historical values, see the International Earth Rotation and Reference Systems Service Web site (`http://www.iers.org`) and navigate to the Earth Orientation Data Data/Products page.
Fifth (Optional)	Scalar	Time increment, for example the Clock block. If the **Higher accuracy parameters** check box is cleared and the **Time Increment** parameter is a value other than `None`, the block has no input.

Output	Dimension Type	Description
First	3-by-3 array	Direction cosine or transformation matrix.

See Also

ECEF Position to LLA | Geocentric to Geodetic Latitude | Geodetic to Geocentric Latitude | LLA to ECEF Position

Topics

http://www.iers.org

Introduced in R2013b

Direction Cosine Matrix to Rodrigues

Convert direction cosine matrix to Euler-Rodrigues vector
Library: Utilities / Axes Transformations

Description

The Direction Cosine Matrix to Rodrigues block determines the 3-by-3 direction cosine matrix from a 3-element Euler-Rodrigues vector.

Ports

Input

`DCM` — Direction cosine matrix
3-by-3 matrix

Direction cosine matrix from which to determine the Euler-Rodrigues vector.

Data Types: `double`

Output

`rod` — Euler-Rodrigues vector
3-element vector

Euler-Rodrigues vector determined from the direction cosine matrix.

Data Types: `double`

Parameters

`Action for invalid DCM` — Block behavior
None (default) | scalar

Block behavior when direction cosine matrix is invalid (not orthogonal).

- Warning — Displays warning and indicates that the direction cosine matrix is invalid.
- Error — Displays error and indicates that the direction cosine matrix is invalid.
- None — Does not display warning or error (default).

Data Types: `char` | `string`

`Tolerance for DCM validation` — Tolerance
`eps(2)` (default) | scalar

Tolerance of direction cosine matrix validity, specified as a scalar. The block considers the direction cosine matrix valid if these conditions are true:

- The transpose of the direction cosine matrix times itself equals 1 within the specified tolerance (`transpose(n)*n == 1±tolerance`)
- The determinant of the direction cosine matrix equals 1 within the specified tolerance (`det(n) == 1±tolerance`).

Data Types: `double`

Algorithms

An Euler-Rodrigues vector $\vec{b}$ represents a rotation by integrating a direction cosine of a rotation axis with the tangent of half the rotation angle as follows:

$$\vec{b} = \begin{bmatrix} b_x & b_y & b_z \end{bmatrix}$$

where:

$$b_x = \tan\left(\frac{1}{2}\theta\right)s_x,$$
$$b_y = \tan\left(\frac{1}{2}\theta\right)s_y,$$
$$b_z = \tan\left(\frac{1}{2}\theta\right)s_z$$

are the Rodrigues parameters. Vector $\vec{s}$ represents a unit vector around which the rotation is performed. Due to the tangent, the rotation vector is indeterminate when the rotation angle equals ±pi radians or ±180 deg. Values can be negative or positive.

References

[1] Dai, J.S. "Euler-Rodrigues formula variations, quaternion conjugation and intrinsic connections." *Mechanism and Machine Theory*, 92, 144-152. Elsevier, 2015.

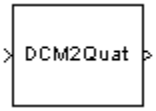
See Also

Quaternions to Rodrigues | Rodrigues to Direction Cosine Matrix | Rodrigues to Quaternions | Rodrigues to Rotation Angles | Rotation Angles to Rodrigues

Introduced in R2017a

Direction Cosine Matrix to Quaternions

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 that 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.

Parameters

Action for invalid DCM

Block behavior when direction cosine matrix is invalid (not orthogonal).

- Warning — Displays warning and indicates that the direction cosine matrix is invalid.
- Error — Displays error and indicates that the direction cosine matrix is invalid.
- None — Does not display warning or error (default).

Tolerance for DCM validation

Tolerance of direction cosine matrix validity, specified as a scalar. Default is `eps(2)`. The block considers the direction cosine matrix valid if these conditions are true:

- The transpose of the direction cosine matrix times itself equals `1` within the specified tolerance (`transpose(n)*n == 1±tolerance`)
- The determinant of the direction cosine matrix equals `1` within the specified tolerance (`det(n) == 1±tolerance`).

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

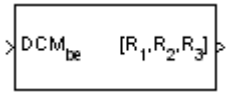
Quaternions to Direction Cosine Matrix

Quaternions to Rotation Angles

Introduced before R2006a

Direction Cosine Matrix to Rotation Angles

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).

Parameters

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.

Rotation order for three wind rotation angles.

For the 'ZYX', 'ZXY', 'YXZ', 'YZX', 'XYZ', and 'XZY' rotations, the block generates an R2 angle that lies between ±pi/2 radians, and R1 and R3 angles that lie between ±pi radians.

For the 'ZYZ', 'ZXZ', 'YXY', 'YZY', 'XYX', and 'XZX' rotations, the block generates an R2 angle that lies between 0 and pi radians, and R1 and R3 angles that lie between ±pi radians. However, in the latter case, when R2 is 0 ±pi radians, R3 is set to 0 radians.

Action for invalid DCM

Block behavior when direction cosine matrix is invalid (not orthogonal).

- Warning — Displays warning and indicates that the direction cosine matrix is invalid.
- Error — Displays error and indicates that the direction cosine matrix is invalid.
- None — Does not display warning or error (default).

Tolerance for DCM validation

Tolerance of direction cosine matrix validity, specified as a scalar. Default is `eps(2)`. The block considers the direction cosine matrix valid if these conditions are true:

- The transpose of the direction cosine matrix times itself equals `1` within the specified tolerance (`transpose(n)*n == 1±tolerance`)
- The determinant of the direction cosine matrix equals `1` within the specified tolerance (`det(n) == 1±tolerance`).

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

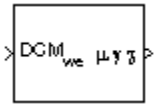
Quaternions to Direction Cosine Matrix

Rotation Angles to Direction Cosine Matrix

Introduced in R2007b

Direction Cosine Matrix to Wind Angles

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 (χ) to axes (ox_1, oy_1, oz_1)
2. A rotation about oy_1 through the flight path angle (γ) to axes (ox_2, oy_2, oz_2)
3. A rotation about ox_2 through the bank angle (μ) 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:

$$\mu = \text{atan}\left(\frac{DCM(2,3)}{DCM(3,3)}\right)$$

$$\gamma = \text{asin}\left(-DCM(1,3)\right)$$

$$\chi = \text{atan}\left(\frac{DCM(1,2)}{DCM(1,1)}\right)$$

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 ±90 degrees, and bank and heading angles that lie between ±180 degrees.

See Also

Direction Cosine Matrix Body to Wind

Direction Cosine Matrix Body to Wind to Alpha and Beta

Direction Cosine Matrix to Rotation Angles

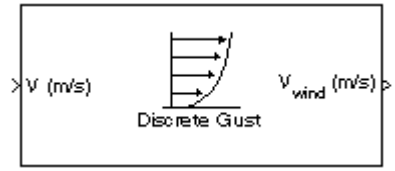
Rotation Angles to Direction Cosine Matrix

Wind Angles to Direction Cosine Matrix

Introduced before R2006a

Discrete Wind Gust Model

Generate discrete wind gust



Library

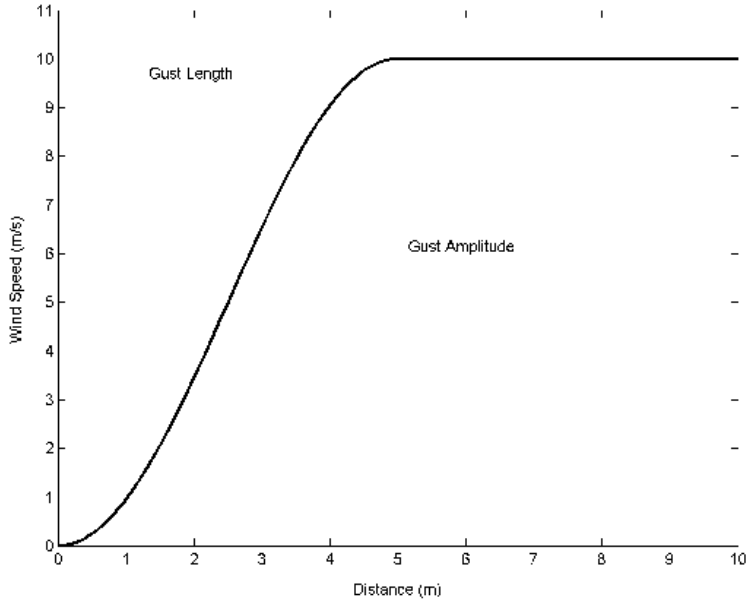
Environment/Wind

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.



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

$$V_{wind} = \begin{cases} 0 & x < 0 \\ \frac{V_m}{2}\left(1 - \cos\left(\frac{\pi x}{d_m}\right)\right) & 0 \le x \le d_m \\ V_m & x > d_m \end{cases}$$

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.

Parameters

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.

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)

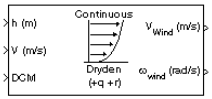
Von Karman Wind Turbulence Model (Continuous)

Wind Shear Model

Introduced before R2006a

Dryden Wind Turbulence Model (Continuous)

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, Military Handbook MIL-HDBK-1797, Military Handbook MIL-HDBK-1797B.

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 Ω radians per meter, the circular frequency ω is calculated by multiplying V by Ω. MIL-F-8785C and MIL-HDBK-1797/1797B provide these definitions of longitudinal, lateral, and vertical component spectra functions:

	MIL-F-8785C	**MIL-HDBK-1797 and MIL-HDBK-1797B**
Longitudinal		
$\Phi_u(\omega)$	$\frac{2\sigma_u^2 L_u}{\pi V}\cdot\frac{1}{1+\left(L_u\frac{\omega}{V}\right)^2}$	$\frac{2\sigma_u^2 L_u}{\pi V}\cdot\frac{1}{1+\left(L_u\frac{\omega}{V}\right)^2}$

	MIL-F-8785C	MIL-HDBK-1797 and MIL-HDBK-1797B
$\Phi_{p_g}(\omega)$	$\frac{\sigma_w^2}{VL_w}\cdot\frac{0.8\left(\frac{\pi L_w}{4b}\right)^{1/3}}{1+\left(\frac{4b\omega}{\pi V}\right)^2}$	$\frac{\sigma_w^2}{2VL_w}\cdot\frac{0.8\left(\frac{2\pi L_w}{4b}\right)^{1/3}}{1+\left(\frac{4b\omega}{\pi V}\right)^2}$
Lateral		
$\Phi_v(\omega)$	$\frac{\sigma_v^2 L_v}{\pi V}\cdot\frac{1+3\left(L_v\frac{\omega}{V}\right)^2}{\left[1+\left(L_v\frac{\omega}{V}\right)^2\right]^2}$	$\frac{2\sigma_v^2 L_v}{\pi V}\cdot\frac{1+12\left(L_v\frac{\omega}{V}\right)^2}{\left[1+4\left(L_v\frac{\omega}{V}\right)^2\right]^2}$
$\Phi_r(\omega)$	$\frac{\mp\left(\frac{\omega}{V}\right)^2}{1+\left(\frac{3b\omega}{\pi V}\right)^2}\cdot\Phi_v(\omega)$	$\frac{\mp\left(\frac{\omega}{V}\right)^2}{1+\left(\frac{3b\omega}{\pi V}\right)^2}\cdot\Phi_v(\omega)$
Vertical		
$\Phi_w(\omega)$	$\frac{\sigma_w^2 L_w}{\pi V}\cdot\frac{1+3\left(L_w\frac{\omega}{V}\right)^2}{\left[1+\left(L_w\frac{\omega}{V}\right)^2\right]^2}$	$\frac{2\sigma_w^2 L_w}{\pi V}\cdot\frac{1+12\left(L_w\frac{\omega}{V}\right)^2}{\left[1+4\left(L_w\frac{\omega}{V}\right)^2\right]^2}$
$\Phi_q(\omega)$	$\frac{\pm\left(\frac{\omega}{V}\right)^2}{1+\left(\frac{4b\omega}{\pi V}\right)^2}\cdot\Phi_w(\omega)$	$\frac{\pm\left(\frac{\omega}{V}\right)^2}{1+\left(\frac{4b\omega}{\pi V}\right)^2}\cdot\Phi_w(\omega)$

where:

- b represents the aircraft wingspan.
- L_u, L_v, L_w represent the turbulence scale lengths.
- σ_u, σ_v, σ_w represent the turbulence intensities.

The spectral density definitions of turbulence angular rates are defined in the specifications as three variations:

$$p_g = \frac{\partial w_g}{\partial y} \qquad q_g = \frac{\partial w_g}{\partial x} \qquad r_g = -\frac{\partial v_g}{\partial x}$$

$$p_g = \frac{\partial w_g}{\partial y} \qquad q_g = \frac{\partial w_g}{\partial x} \qquad r_g = \frac{\partial v_g}{\partial x}$$

$$p_g = -\frac{\partial w_g}{\partial y} \qquad q_g = -\frac{\partial w_g}{\partial x} \qquad r_g = \frac{\partial v_g}{\partial x}$$

The variations affect only the vertical (q_g) and lateral (r_g) turbulence angular rates.

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. The variations exist because the turbulence angular rate spectra contribute less to the aircraft gust response than the turbulence velocity.

The variations result in these combinations of vertical and lateral turbulence angular rate spectra.

Vertical	Lateral
$\Phi_q(\omega)$	$-\Phi_r(\omega)$
$\Phi_q(\omega)$	$\Phi_r(\omega)$
$-\Phi_q(\omega)$	$\Phi_r(\omega)$

To generate a signal with correct characteristics, a band-limited white noise signal is passed through forming filters. The forming filters are derived from the spectral square roots of the spectrum equations.

MIL-F-8785C and MIL-HDBK-1797/1797B provide these transfer functions:

	MIL-F-8785C	MIL-HDBK-1797 and MIL-HDBK-1797B
Longitudinal		
$H_u(s)$	$\sigma_u\sqrt{\frac{2L_u}{\pi V}}\cdot\frac{1}{1+\frac{L_u}{V}s}$	$\sigma_u\sqrt{\frac{2L_u}{\pi V}}\cdot\frac{1}{1+\frac{L_u}{V}s}$
$H_p(s)$	$\sigma_w\sqrt{\frac{0.8}{V}}\cdot\frac{\left(\frac{\pi}{4b}\right)^{1/6}}{L_w^{1/3}\left(1+\left(\frac{4b}{\pi V}\right)s\right)}$	$\sigma_w\sqrt{\frac{0.8}{V}}\cdot\frac{\left(\frac{\pi}{4b}\right)^{1/6}}{(2L_w)^{1/3}\left(1+\left(\frac{4b}{\pi V}\right)s\right)}$
Lateral		
$H_v(s)$	$\sigma_v\sqrt{\frac{L_v}{\pi V}}\cdot\frac{1+\frac{\sqrt{3}L_v}{V}s}{\left(1+\frac{L_v}{V}s\right)^2}$	$\sigma_v\sqrt{\frac{2L_v}{\pi V}}\cdot\frac{1+\frac{2\sqrt{3}L_v}{V}s}{\left(1+\frac{2L_v}{V}s\right)^2}$
$H_r(s)$	$\frac{\mp\frac{s}{V}}{\left(1+\left(\frac{3b}{\pi V}\right)s\right)}\cdot H_v(s)$	$\frac{\mp\frac{s}{V}}{\left(1+\left(\frac{3b}{\pi V}\right)s\right)}\cdot H_v(s)$
Vertical		
$H_w(s)$	$\sigma_w\sqrt{\frac{L_w}{\pi V}}\cdot\frac{1+\frac{\sqrt{3}L_w}{V}s}{\left(1+\frac{L_w}{V}s\right)^2}$	$\sigma_w\sqrt{\frac{2L_w}{\pi V}}\cdot\frac{1+\frac{2\sqrt{3}L_w}{V}s}{\left(1+\frac{2L_w}{V}s\right)^2}$

	MIL-F-8785C	MIL-HDBK-1797 and MIL-HDBK-1797B
$H_q(s)$	$\frac{\pm\frac{s}{V}}{\left(1+\left(\frac{4b}{\pi V}\right)s\right)}\cdot H_w(s)$	$\frac{\pm\frac{s}{V}}{\left(1+\left(\frac{4b}{\pi V}\right)s\right)}\cdot H_w(s)$

Divided into two distinct regions, the turbulence scale lengths and intensities are functions of altitude.

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 Under 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 and MIL-HDBK-1797B
$L_w = h$ $L_u = L_v = \frac{h}{(0.177+0.000823h)^{1.2}}$	$2L_w = h$ $L_u = 2L_v = \frac{h}{(0.177+0.000823h)^{1.2}}$

Typically, at 20 feet (6 meters) the wind speed is 15 knots in light turbulence, 30 knots in moderate turbulence, and 45 knots for severe turbulence. See these turbulence intensities, where W_{20} is the wind speed at 20 feet (6 meters).

$$\sigma_w = 0.1W_{20}$$
$$\frac{\sigma_u}{\sigma_w} = \frac{\sigma_v}{\sigma_w} = \frac{1}{(0.177+0.000823h)^{0.4}}$$

The turbulence axes orientation in this region is defined:

- 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 Above 2000 Feet)

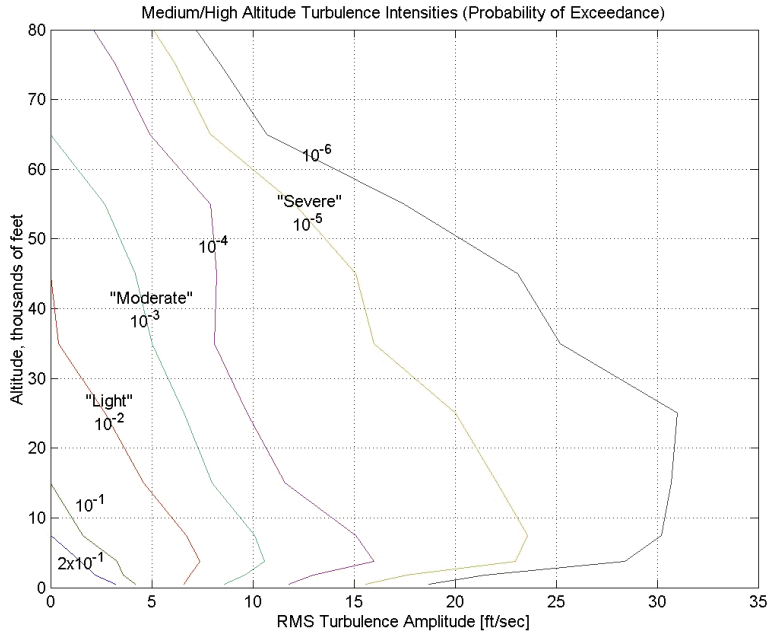
Turbulence scale lengths and intensities for medium-to-high altitudes the are based on the assumption that the turbulence is isotropic. MIL-F-8785C and MIL-HDBK-1797/1797B provide these representations of scale lengths:

MIL-F-8785C	MIL-HDBK-1797 and MIL-HDBK-1797B
$L_u = L_v = L_w = 1750$ 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.



Between Low and Medium/High Altitudes (Between 1000 and 2000 Feet)

At altitudes between 1000 and 2000, 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.

Parameters

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

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.
`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

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)

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 exceeding the turbulence intensity.

Scale length at medium/high altitudes (m)

Turbulence scale length above 2000 feet, assumed constant. MIL-F-8785C and MIL-HDBK-1797/1797B recommend 1750 feet for the longitudinal turbulence scale length of the Dryden spectra.

Note An alternative scale length value changes the power spectral density asymptote and gust load.

Wingspan

Wingspan 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

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 this parameter generates the turbulence signals.

Ports

Input	Dimension Type	Description
First	scalar	Contains the altitude, in units selected.
Second	scalar	Contains the aircraft speed, in units selected.
Third	3-by-3 matrix	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.

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 ground speed.

The turbulence model describes an average of all conditions for clear air turbulence. These factors are not incorporated into the model:

- Terrain roughness
- Lapse rate
- Wind shears
- Mean wind magnitude
- Other meteorological factors

Examples

See Airframe/Environment Models/Wind Models in `aeroblk_HL20` for an example of this block.

References

Chalk, Charles, T.P. Neal, T.M. Harris, Francis E. Pritchard, and Robert J. Woodcock. *Background Information and User Guide for MIL-F-8785B(ASG), "Military Specification-Flying Qualities of Piloted Airplanes."* AD869856. Buffalo, NY: Cornell Aeronautical Laboratory, 1969.

Flying Qualities of Piloted Aircraft. Department of Defense Handbook. MIL-HDBK-1797. Washington, DC: U.S. Department of Defense, 1997.

Flying Qualities of Piloted Aircraft. Department of Defense Handbook. MIL-HDBK-1797B. Washington, DC: U.S. Department of Defense, 2012.

Flying Qualities of Piloted Airplanes. U.S. Military Specification MIL-F-8785C. Washington, D.C.: U.S. Department of Defense, 1980.

Hoblit, F., *Gust Loads on Aircraft: Concepts and Applications*, AIAA Education Series, 1988.

Ly, U. and Y. Chan. "Time-Domain Computation of Aircraft Gust Covariance Matrices," AIAA Paper 80-1615, presented at the 6th Atmospheric Flight Mechanics Conference, Danvers, Massachusetts, August 1980.

McFarland, Richard E, A Standard Kinematic Model for Flight Simulation at NASA-AMES. NASA CR-2497. Mountain view, CA: Computer Sciences Corporation, 1975.

McRuer, Duane, Dunstan Graham, and Irving Ashkenas. *Aircraft Dynamics and Automatic Control* Princeton University Press, 1974, R1990.

Moorhouse, David J. and Robert J. Woodcock. *Background Information and User Guide for MIL-F-8785C, "Military Specification—Flying Qualities of Piloted Airplanes."* ADA119421. Wright-Patterson AFB, OH: Air Force Wright Aeronautical Labs, 1982.

Tatom, Frank B., George H. Fichtl, and Stephen R. Smith. “Simulation of Atmospheric Turbulent Gusts and Gust Gradients,” AIAA Paper 81-0300, presented at the 19th Aerospace Sciences Meeting, St. Louis, Missouri, January 1981.

Yeager, Jessie, *Implementation and Testing of Turbulence Models for the F18-HARV Simulation* NASA CR-1998-206937. Hampton, VA: Lockheed Martin Engineering & Sciences, 1998.

See Also

Dryden Wind Turbulence Model (Discrete)

Discrete Wind Gust Model

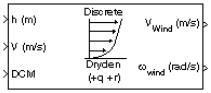
Wind Shear Model

Von Karman Wind Turbulence Model (Continuous)

Introduced before R2006a

Dryden Wind Turbulence Model (Discrete)

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, Military Handbook MIL-HDBK-1797, and Military Handbook MIL-HDBK-1797B.

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 Ω radians per meter, the circular frequency ω is calculated by multiplying V by Ω. The following table displays the component spectra functions:

	MIL-F-8785C	**MIL-HDBK-1797 and MIL-HDBK-1797B**
Longitudinal		
$\Phi_u(\omega)$	$\frac{2\sigma_u^2 L_u}{\pi V}\cdot\frac{1}{1+\left(L_u\frac{\omega}{V}\right)^2}$	$\frac{2\sigma_u^2 L_u}{\pi V}\cdot\frac{1}{1+\left(L_u\frac{\omega}{V}\right)^2}$

	MIL-F-8785C	MIL-HDBK-1797 and MIL-HDBK-1797B
$\Phi_p(\omega)$	$\frac{\sigma_w^2}{VL_w}\cdot\frac{0.8\left(\frac{\pi L_w}{4b}\right)^{1/3}}{1+\left(\frac{4b\omega}{\pi V}\right)^2}$	$\frac{\sigma_w^2}{2VL_w}\cdot\frac{0.8\left(\frac{2\pi L_w}{4b}\right)^{1/3}}{1+\left(\frac{4b\omega}{\pi V}\right)^2}$
Lateral		
$\Phi_v(\omega)$	$\frac{\sigma_v^2 L_v}{\pi V}\cdot\frac{1+3\left(L_v\frac{\omega}{V}\right)^2}{\left[1+\left(L_v\frac{\omega}{V}\right)^2\right]^2}$	$\frac{2\sigma_v^2 L_v}{\pi V}\cdot\frac{1+12\left(L_v\frac{\omega}{V}\right)^2}{\left[1+4\left(L_v\frac{\omega}{V}\right)^2\right]^2}$
$\Phi_r(\omega)$	$\frac{\mp\left(\frac{\omega}{V}\right)^2}{1+\left(\frac{3b\omega}{\pi V}\right)^2}\cdot\Phi_v(\omega)$	$\frac{\mp\left(\frac{\omega}{V}\right)^2}{1+\left(\frac{3b\omega}{\pi V}\right)^2}\cdot\Phi_v(\omega)$
Vertical		
$\Phi_w(\omega)$	$\frac{\sigma_w^2 L_w}{\pi V}\cdot\frac{1+3\left(L_w\frac{\omega}{V}\right)^2}{\left[1+\left(L_w\frac{\omega}{V}\right)^2\right]^2}$	$\frac{2\sigma_w^2 L_w}{\pi V}\cdot\frac{1+12\left(L_w\frac{\omega}{V}\right)^2}{\left[1+4\left(L_w\frac{\omega}{V}\right)^2\right]^2}$
$\Phi_q(\omega)$	$\frac{\pm\left(\frac{\omega}{V}\right)^2}{1+\left(\frac{4b\omega}{\pi V}\right)^2}\cdot\Phi_w(\omega)$	$\frac{\pm\left(\frac{\omega}{V}\right)^2}{1+\left(\frac{4b\omega}{\pi V}\right)^2}\cdot\Phi_w(\omega)$

The variable b represents the aircraft wingspan. The variables L_u, L_v, L_w represent the turbulence scale lengths. The variables σ_u, σ_v, σ_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:

$$p_g = \frac{\partial w_g}{\partial y} \qquad q_g = \frac{\partial w_g}{\partial x} \qquad r_g = -\frac{\partial v_g}{\partial x}$$

$$p_g = \frac{\partial w_g}{\partial y} \qquad q_g = \frac{\partial w_g}{\partial x} \qquad r_g = \frac{\partial v_g}{\partial x}$$

$$p_g = -\frac{\partial w_g}{\partial y} \qquad q_g = -\frac{\partial w_g}{\partial x} \qquad r_g = \frac{\partial v_g}{\partial x}$$

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:

Vertical	Lateral
$\Phi_q(\omega)$	$-\Phi_r(\omega)$
$\Phi_q(\omega)$	$\Phi_r(\omega)$
$-\Phi_q(\omega)$	$\Phi_r(\omega)$

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:

	MIL-F-8785C	MIL-HDBK-1797 and MIL-HDBK-1797B
Longitudinal		

	MIL-F-8785C	MIL-HDBK-1797 and MIL-HDBK-1797B
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	$\left(1-\frac{2.6}{\sqrt{L_w b}}T\right)p_g+\left(\sqrt{2\frac{2.6}{\sqrt{L_w b}}T}\right)\left(\frac{\frac{0.95}{\sqrt[3]{2L_w b^2}}}{\sigma_\eta}\eta_4\right)\sigma_w$	MIL-HDBK-1797 $\left(1-\frac{2.6}{\sqrt{2L_w b}}T\right)p_g+\left(\sqrt{2\frac{2.6}{\sqrt{2L_w b}}T}\right)\left(\frac{\frac{1.9}{\sqrt{2L_w b}}}{\sigma_\eta}\eta_4\right)\sigma_w$ MIL-HDBK-1797B $\left(1-\frac{2.6V}{\sqrt{2L_w b}}T\right)p_g+\left(\sqrt{2\frac{2.6V}{\sqrt{2L_w b}}T}\right)\left(\frac{\frac{1.9}{\sqrt{2L_w b}}}{\sigma_\eta}\eta_4\right)\sigma_w$
Lateral		
v_g	$\left(1-\frac{V}{L_u}T\right)v_g+\sqrt{2\frac{V}{L_u}T}\frac{\sigma_v}{\sigma_\eta}\eta_2$	$\left(1-\frac{V}{L_u}T\right)v_g+\sqrt{2\frac{V}{L_u}T}\frac{\sigma_v}{\sigma_\eta}\eta_2$
r_g	$\left(1-\frac{\pi V}{3b}T\right)r_g\mp\frac{\pi}{3b}\left(v_g-v_{g_{past}}\right)$	$\left(1-\frac{\pi V}{3b}T\right)r_g\mp\frac{\pi}{3b}\left(v_g-v_{g_{past}}\right)$
Vertical		

	MIL-F-8785C	MIL-HDBK-1797 and MIL-HDBK-1797B
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	$\left(1-\frac{\pi V}{4b}T\right)q_g\pm\frac{\pi}{4b}\left(w_g-w_{g_{past}}\right)$	$\left(1-\frac{\pi V}{4b}T\right)q_g\pm\frac{\pi}{4b}\left(w_g-w_{g_{past}}\right)$

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 and MIL-HDBK-1797B
$L_w=h$ $L_u=L_v=\frac{h}{(0.177+0.000823h)^{1.2}}$	$2L_w=h$ $L_u=2L_v=\frac{h}{(0.177+0.000823h)^{1.2}}$

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.

At this altitude range, the output of the block is transformed into body coordinates.

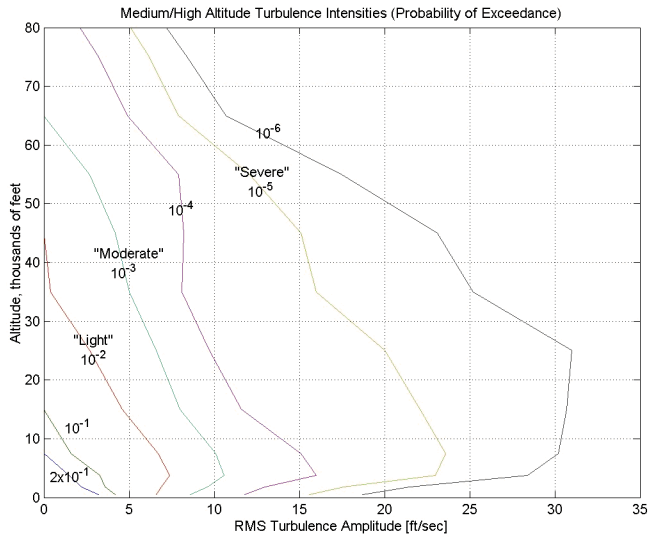
Medium/High Altitudes (Altitude > 2000 feet)

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 and MIL-HDBK-1797B
$L_u = L_v = L_w = 1750$ 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.



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.

Parameters

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

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.
`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 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.

Inputs and Outputs

Input	Dimension Type	Description
First	scalar	Contains the altitude, in units selected.
Second	scalar	Contains the aircraft speed, in units selected.
Third	3-by-3 matrix	Contains the NED direction cosine matrix.

Output	Dimension Type	Description
First	Three-element signal	Contains the turbulence velocities, in the selected units.

Output	Dimension Type	Description
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 factions (except altitude)

References

U.S. Military Handbook MIL-HDBK-1797B, 9 April 2012.

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.

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)

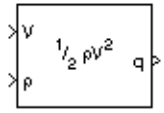
Discrete Wind Gust Model

Wind Shear Model

Introduced before R2006a

Dynamic Pressure

Compute dynamic pressure using velocity and air density



Library

Flight Parameters

Description

The Dynamic Pressure block computes dynamic pressure.

Dynamic pressure is defined as

$$\overline{q} = \frac{1}{2}\rho V^2$$

where ρ is air density and V is velocity.

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

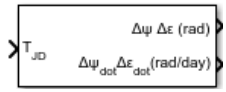
Aerodynamic Forces and Moments

Mach Number

Introduced before R2006a

Earth Nutation

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 Julian date input for the block, you can use the Julian Date Conversion block or the Aerospace Toolbox `juliandate` function to calculate the Julian date. You can also calculate your own Julian date and input it using the Constant block.

Parameters

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.

- `T0 and elapsed Julian time`

  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	Released in 1998. This ephemerides takes into account the Julian date range 2305424.50 (December 9, 1599) to 2525008.50 (February 20, 2201). This block implements these ephemerides with respect to the International Celestial Reference Frame version 1.0, adopted in 1998.
DE421	Released in 2008. This ephemerides takes into account the Julian date range 2414992.5 (December 4, 1899) to 2469808.5 (January 2, 2050). This block implements these ephemerides with respect to the International Celestial Reference Frame version 1.0, adopted in 1998.
DE423	Released in 2010. This ephemerides takes into account the Julian date range 2378480.5 (December 16, 1799) to 2524624.5 (February 1, 2200). This block implements these ephemerides with respect to the International Celestial Reference Frame version 2.0, adopted in 2010.

Ephemerides Model	Description
`DE430`	Released in 2013. This ephemerides takes into account the Julian date range 2287184.5 (December 21, 1549) to 2688976.5 (January 25, 2650). This block implements these ephemerides with respect to the International Celestial Reference Frame version 2.0, adopted in 2010.

Note This block requires that you download ephemeris data with the Add-On Explorer. Click the **Get data** button to start the Add-On Explorer. For more information, see `aeroDataPackage`. If the data is installed, the **Get data** button does not appear.

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	• Julian date (T_{JD}) (default) — One input. Specify a date between the minimum and maximum Julian date. • Fixed Julian date ($T0_{JD}$) plus the elapsed Julian time (ΔT_{JD}) between the fixed date and the ephemeris time. — Two inputs, where the first input is $T0_{JD}$ and the second input is ΔT_{JD}. ΔT_{JD} must be a positive number. The sum of $T0_{JD}$ and ΔT_{JD} must fall between the minimum and maximum Julian date. The block **Epoch** parameter controls the number of block inputs. See the **Ephemerides** parameter for the minimum and maximum Julian dates.
Second (Optional)	Scalar	ΔT_{JD} — Elapsed Julian time (ΔT_{JD}) between the fixed date and the ephemeris time. The sum of $T0_{JD}$ and ΔT_{JD} must fall between the minimum and maximum Julian date. See the **Ephemerides** parameter for the minimum and maximum Julian date.

Output	Dimension Type	Description
First	Vector	Earth nutation in longitude ($\Delta\psi$) and obliquity ($\Delta\varepsilon$). Units are radians.
Second (Optional)	Vector	Earth nutation angular rate for the longitude ($\Delta\psi_{dot}$) and obliquity ($\Delta\varepsilon_{dot}$). 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

Direction Cosine Matrix ECI to ECEF | Julian Date Conversion | Moon Libration | Planetary Ephemeris

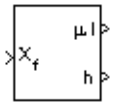
External Websites

`http://ssd.jpl.nasa.gov/?planet_eph_export`

Introduced in R2013a

ECEF Position to LLA

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

$$\iota = \operatorname{atan}\left(\frac{p_y}{p_x}\right)$$

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:

$$\bar{\beta} = \operatorname{atan}\left(\frac{p_z}{(1-f)s}\right)$$

$$\bar{\mu} = \operatorname{atan}\left(\frac{p_z + \frac{e^2(1-f)}{(1-e^2)}R(\sin\beta)^3}{s - e^2R(\cos\beta)^3}\right)$$

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 $\left(\bar{\beta}\right)$ is recalculated using

$$\beta = \operatorname{atan}\left(\frac{(1-f)\sin\mu}{\cos\mu}\right)$$

and geodetic latitude $\left(\bar{\mu}\right)$ is reevaluated. This last step is repeated until $\bar{\mu}$ converges.

The altitude $\left(\bar{h}\right)$ above the planetary ellipsoid is calculated with

$$h = s\cos\mu + \left(p_z + e^2N\sin\mu\right)\sin\mu - N$$

where the radius of curvature in the vertical prime $\left(\bar{N}\right)$ is given by

$$N = \frac{R}{\sqrt{1-e^2(\sin\mu)^2}}$$

Parameters

Units

Specifies the parameter and output units:

Units	Position	Equatorial Radius	Altitude
`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

Input	Dimension Type	Description
First	3-by-1 vector	Contains the position in ECEF frame.

Output	Dimension Type	Description
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.

Assumptions and Limitations

This implementation generates a geodetic latitude that lies between ±90 degrees, and longitude that lies between ±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 2-10.

Direction Cosine Matrix ECEF to NED

Direction Cosine Matrix ECEF to NED to Latitude and Longitude

Geocentric to Geodetic Latitude

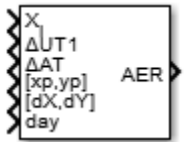
LLA to ECEF Position

Radius at Geocentric Latitude

Introduced before R2006a

ECI Position to AER

Convert Earth-centered inertial (ECI) coordinates to azimuth coordinates



Library

Utilities/Axes Transformations

Description

The ECI Position to AER block converts Earth-centered inertial (ECI) position coordinates to azimuth, elevation, and slant-range coordinates (AER), based on the geodetic position (latitude, longitude, and altitude).

- Azimuth (A) — Angle measured clockwise from true north. It ranges from 0 to 360 degrees.
- Elevation (E) — Angle between a plane perpendicular to the ellipsoid and the line that goes from the local reference to the object position. It ranges from –90 to 90 degrees.
- Slant range (R) — Straight line distance between the local reference and the object.

Parameters

Reduction

Reduction method to convert the coordinates. Select either:

- `IAU-76/FK5`

  Reduce the calculation using the International Astronomical Union 76/Fifth Fundamental Catalogue (IAU-76/FK5) reference system. Choose this reduction method if the reference coordinate system for the conversion is FK5.

Note This method uses the IAU 1976 precession model and the IAU 1980 theory of nutation to reduce the calculation. This model and theory are no longer current, but the software provides this reduction method for existing implementations. Because of the polar motion approximation that this reduction method uses, the block calculates the transformation matrix rather than the direction cosine matrix.

- `IAU-2000/2006`

  Reduce the calculation using the International Astronomical Union 2000/2006 reference system. Choose this reduction method if the reference coordinate system for the conversion is IAU-2000. This reduction method uses the P03 precession model to reduce the calculation.

Year

Specify the year used to calculate the Universal Coordinated Time (UTC) date. Enter a double value that is a whole number greater than 1, such as `2013`.

Month

Specify the month used to calculate the UTC date. From the list, select the month from `January` to `December`.

Day

Specify the day used to calculate the UTC date. From the list, select the day from `1` to `31`.

Hour

Specify the hour used to calculate the UTC date. Enter a double value that is a whole number from `0` to `24`.

Minutes

Specify the minutes used to calculate the UTC date. Enter a double value that is a whole number from `0` to `60`.

Seconds

Specify the seconds used to calculate the UTC date. Enter a double value that is a whole number from `0` to `60`.

Time Increment

Specify the time increment between the specified date and the desired model simulation time. The block adjusts the calculated direction cosine matrix to take into account the time increment from model simulation. For example, selecting `Day` and connecting a simulation timer to the port means that each time increment unit is one day. The block adjusts the calculation based on that simulation time.

This parameter corresponds to the fifth block input, the clock source.

Possible values are `Day`, `Hour`, `Min`, `Sec`, and `None`. If you select `None`, the calculated Julian date does not take into account the model simulation time. Selecting `None` removes the fifth block input.

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.

Higher accuracy parameters

Select this check box to allow the following as block inputs. These inputs let you better control the conversion result. See “Inputs and Outputs” on page 4-287 for a description.

X_i
$\Delta UT1$
ΔAT
$[xp,yp]$
$[\Delta\delta\psi, \Delta\delta\varepsilon]$ or $[dX,dY]$
day

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 **Earth model** is set to `WGS84`.

Earth model

Specifies the planet model to use: `Custom` or `WGS84`.

Flattening

Specifies the flattening of the planet. This option is only available with **Earth model** set to `Custom`.

Equatorial radius

Specifies the radius of the planet at its equator. This option is only available with **Earth model** set to `Custom`.

Initial geodetic latitude and longitude [deg]

Specifies the reference location in latitude and longitude in degrees.

Angular direction of the local reference system (degrees clockwise from north)

Specifies angle for converting the flat Earth *x* and *y* coordinates to north and east coordinates, respectively. An example is the angle between the vessel and the true geodetic north.

Reference height

Specifies the reference height measured from the surface of the Earth to the flat Earth frame. It uses the same units as the ECI position. Estimate the reference height relative to the Earth frame.

Inputs and Outputs

Input	Dimension Type	Description
First	3-by-1 element vector	*X*, position in ECI coordinates.
Second (Optional)	Scalar	Δ*UT1*, difference between UTC and Universal Time (UT1), in seconds, for which the function calculates the direction cosine or transformation matrix, for example, `0.234`.
Third (Optional)	Scalar	Δ*AT*, difference between International Atomic Time (IAT) and UTC, in seconds, for which the function calculates the direction cosine or transformation matrix, for example, `32`.
Fourth (Optional)	1-by-2 array	[*xp*,*yp*], polar displacement of the Earth, in radians, from the motion of the Earth crust, along the *x*- and *y*-axes, for example, `[-0.0682e-5 0.1616e-5]`.

Input	Dimension Type	Description
Fifth (Optional)	1-by-2 array	• If reduction method is `IAU-2000/2006`, this input is the adjustment to the location of the Celestial Intermediate Pole (CIP), specified in radians. This location ([d*X*,d*Y*]) is along the *x*- and *y*-axes, for example, `[-0.2530e-6 -0.0188e-6]`. • If reduction method is `IAU-76/FK5`, this input is the adjustment to the longitude ([Δδψ, Δδε]), specified in radians, for example, `[-0.2530e-6 -0.0188e-6]`. For historical values, see the International Earth Rotation and Reference Systems Service website (`http://www.iers.org`) and navigate to the Earth Orientation Data Data/Products page.
Sixth	Scalar	Time increment, for example, the Clock block. If the **Higher accuracy parameters** check box is cleared and the **Time Increment** parameter is a value other than `None`, the block has no input.

Output	Dimension Type	Description
First	3-by-1 element vector	3-by-1 element vector with the local reference coordinates azimuth (degrees), elevation (degrees), and slant range (meters).

See Also

Direction Cosine Matrix ECI to ECEF | ECI Position to LLA | LLA to ECI Position

Topics

http://www.iers.org

Introduced in R2015a

ECI Position to LLA

Convert Earth-centered inertial (ECI) coordinates to geodetic latitude, longitude, altitude (LLA) coordinates

Library

Utilities/Axes Transformations

Description

The ECI Position to LLA block converts Earth-centered inertial (ECI) position coordinates to geodetic latitude, longitude, altitude (LLA) coordinates, based on the specified reduction method and Universal Coordinated Time (UTC), for the specified time and geophysical data.

Parameters

Reduction

Reduction method to convert the coordinates. Select one of the following:

- `IAU-76/FK5`

  Reduce the calculation using the International Astronomical Union (IAU)-76/Fifth Fundamental Catalogue (FK5) (IAU-76/FK5) reference system. Choose this reduction method if the reference coordinate system for the conversion is FK5.

  Note This method uses the IAU 1976 precession model and the IAU 1980 theory of nutation to reduce the calculation. This model and theory are no longer current,

but the software provides this reduction method for existing implementations. Because of the polar motion approximation that this reduction method uses, the block calculates the transformation matrix rather than the direction cosine matrix.

- `IAU-2000/2006`

  Reduce the calculation using the International Astronomical Union (IAU)-2000/2006 reference system. Choose this reduction method if the reference coordinate system for the conversion is IAU-2000. This reduction method uses the P03 precession model to reduce the calculation.

Year

Specify the year used to calculate the Universal Coordinated Time (UTC) date. Enter a double value that is a whole number greater than 1, such as `2013`.

Month

Specify the month used to calculate the UTC date. From the list, select the month from `January` to `December`.

Day

Specify the day used to calculate the UTC date. From the list, select the day from `1` to `31`.

Hour

Specify the hour used to calculate the UTC date. Enter a double value that is a whole number from `0` to `24`.

Minutes

Specify the minutes used to calculate the UTC date. Enter a double value that is a whole number from `0` to `60`.

Seconds

Specify the seconds used to calculate the UTC date. Enter a double value that is a whole number from `0` to `60`.

Time Increment

Specify the time increment between the specified date and the desired model simulation time. The block adjusts the calculated direction cosine matrix to take into account the time increment from model simulation. For example, selecting `Day` and connecting a simulation timer to the port means that each time increment unit is one day. The block adjusts its calculation based on that simulation time.

This parameter corresponds to the fifth block input, the clock source.

Possible values are `Day`, `Hour`, `Min`, `Sec`, and `None`. If you select `None`, the calculated Julian date does not take into account the model simulation time. Selecting this option removes the fifth block input.

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.

Higher accuracy parameters

Select this check box to enable the following inputs. These inputs let you better control the conversion result. See “Inputs and Outputs” on page 4-287 for a description.

X
Δ*UT1*
Δ*AT*
[*xp*,*yp*]
[Δδψ, Δδε] or [d*X*,d*Y*]
day

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)`.

Earth model

Specifies the planet model to use: `Custom` or `WGS84`.

Flattening

Specifies the flattening of the planet. This option is available only with **Earth model** **Custom**.

Equatorial radius

Specifies the radius of the planet at its equator. This option is available only with **Earth model Custom**.

Inputs and Outputs

Input	Dimension Type	Description
First	3-by-1 element vector	*X*, position in ECI coordinates.
Second (Optional)	Scalar	Δ*UT1*, difference between UTC and Universal Time (UT1), in seconds, for which the function calculates the direction cosine or transformation matrix, for example, `0.234`.
Third (Optional)	Scalar	Δ*AT*, difference between International Atomic Time (IAT) and UTC, in seconds, for which the function calculates the direction cosine or transformation matrix, for example, `32`.
Fourth (Optional)	1-by-2 array	[*xp*,*yp*], polar displacement of the Earth, in radians, from the motion of the Earth crust, along the *x*- and *y*-axes, for example, `[-0.0682e-5 0.1616e-5]`.
Fifth (Optional)	1-by-2 array	• If reduction method is `IAU-2000/2006`, this input is the adjustment to the location of the Celestial Intermediate Pole (CIP), specified in radians. This location ([d*X*,d*Y*]) is along the *x*- and *y*-axes, for example, `[-0.2530e-6 -0.0188e-6]`. • If reduction method is `IAU-76/FK5`, this input is the adjustment to the longitude ([Δδψ, Δδε]), specified in radians, for example, `[-0.2530e-6 -0.0188e-6]`. For historical values, see the International Earth Rotation and Reference Systems Service website (`http://www.iers.org`) and navigate to the Earth Orientation Data Data/Products page.

Input	Dimension Type	Description
Sixth	Scalar	Time increment, for example, the Clock block. If the **Higher accuracy parameters** check box is cleared and the **Time Increment** parameter is a value other than `None`, the block has no input.

Output	Dimension Type	Description
First	3-by-1 element vector	Original position vector in geodetic LLA coordinates, in degrees.

See Also

LLA to ECI Position

Topics

http://www.iers.org

Introduced in R2014a

EGM96 Geoid

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.

Parameters

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 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
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

Introduced in R2007b

Exhaust Gas Temperature (EGT) Indicator

Display measurements for engine exhaust gas temperature (EGT)

Library

Flight Instruments

Description

The EGT Indicator block displays temperature measurements for engine exhaust gas temperature (EGT) in Celsius.

This block displays values using both:

- A needle on a gauge. A major tick is (**Maximum**-**Minimum**)/1,000 degrees, a minor tick is (**Maximum**-**Minimum**)/200 degrees Celsius.
- A numeric indicator. The operating range for the indicator goes from **Minimum** to **Maximum** degrees Celsius.

If the value of the signal is under **Minimum**, the needle displays 5 degrees under the **Minimum** value, the numeric display shows the **Minimum** value. If the value exceeds the **Maximum** value, the needle displays 5 degrees over the maximum tick, and the numeric displays the **Maximum** value.

Parameters

Connection

Select a 1-element signal to connect to the block.

To view the data from a signal, select a signal in the model. The signal appears in the **Connection** table. Select the option button next to the signal you want to display. Click **Apply** to connect the signal.

The table has a row for the signal connected to the block. If there are no signals selected in the model, or the block is not connected to any signals, the table is empty.

Minimum

Specify the minimum tick mark value.

Default value is 0.

Specify this number as a finite, real, double, scalar value.

The **Minimum** tick value must be less than the **Maximum** tick value.

Maximum

Specify the maximum tick mark value.

Default value is 1000.

Specify this number as a finite, real, double, scalar value.

The **Minimum** tick value must be greater than the **Maximum** tick value.

Scale Colors

Specify ranges of color bands on the outside of the scale. Specify the minimum and maximum color range to display on the gauge.

Label

Name of connected signal.

- `Top`

  Show label at the top of the block.

- `Bottom`

  Show label at the bottom of the block.

- `Hide`

  Do not show the label or instructional text when the block is not connected.

See Also

Airspeed Indicator | Altimeter | Artificial Horizon | Climb Rate Indicator | Heading Indicator | Revolutions Per Minute (RPM) Indicator | Turn Coordinator

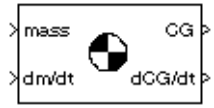
Topics

“Display Measurements with Cockpit Instruments” on page 2-47
“Flight Instrument Gauges” on page 2-46

Introduced in R2016a

Estimate Center of Gravity

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.

Parameters

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

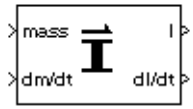
Estimate Inertia Tensor

Moments About CG Due to Forces

Introduced before R2006a

Estimate Inertia Tensor

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.

Parameters

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

Introduced before R2006a

Rodrigues to Direction Cosine Matrix

Convert Euler-Rodrigues vector to direction cosine matrix
Library: Utilities / Axes Transformations

Description

The Rodrigues to Direction Cosine Matrix block determines the 3-by-3 direction cosine matrix from a 3-element Euler-Rodrigues vector.

Ports

Input

`rod` — Euler-Rodrigues vector
3-element vector

Euler-Rodrigues vector from which to determine the direction cosine matrix.

Data Types: `double`

Output

`DCM` — Direction cosine matrix
3-by-3 matrix

Direction cosine matrix determined from the Euler-Rodrigues vector.

Data Types: `double`

Algorithms

An Euler-Rodrigues vector $\vec{b}$ represents a rotation by integrating a direction cosine of a rotation axis with the tangent of half the rotation angle as follows:

$$\vec{b} = \begin{bmatrix} b_x & b_y & b_z \end{bmatrix}$$

where:

$$b_x = \tan\left(\frac{1}{2}\theta\right)s_x,$$
$$b_y = \tan\left(\frac{1}{2}\theta\right)s_y,$$
$$b_z = \tan\left(\frac{1}{2}\theta\right)s_z$$

are the Rodrigues parameters. Vector $\vec{s}$ represents a unit vector around which the rotation is performed. Due to the tangent, the rotation vector is indeterminate when the rotation angle equals ±pi radians or ±180 deg. Values can be negative or positive.

References

[1] Dai, J.S. "Euler-Rodrigues formula variations, quaternion conjugation and intrinsic connections." *Mechanism and Machine Theory*, 92, 144-152. Elsevier, 2015.

See Also

Direction Cosine Matrix to Rodrigues | Quaternions to Rodrigues | Rodrigues to Quaternions | Rodrigues to Rotation Angles | Rotation Angles to Rodrigues

Introduced in R2017a

Rodrigues to Quaternions

Convert Euler-Rodrigues vector to quaternion
Library: Utilities / Axes Transformations

Description

The Rodrigues to Quaternions block determines the 4-by-1 quaternion from a 3-element Euler-Rodrigues vector.

Ports

Input

`rod` — Euler-Rodrigues vector
3-element vector

Euler-Rodrigues vector from which to determine the quaternion.

Data Types: `double`

Output

`q` — Quaternion
4-by-1 matrix

Quaternion determined from the Euler-Rodrigues vector.

Data Types: `double`

Algorithms

An Euler-Rodrigues vector $\vec{b}$ represents a rotation by integrating a direction cosine of a rotation axis with the tangent of half the rotation angle as follows:

$$\vec{b} = \begin{bmatrix} b_x & b_y & b_z \end{bmatrix}$$

where:

$$b_x = \tan\left(\frac{1}{2}\theta\right)s_x,$$
$$b_y = \tan\left(\frac{1}{2}\theta\right)s_y,$$
$$b_z = \tan\left(\frac{1}{2}\theta\right)s_z$$

are the Rodrigues parameters. Vector $\vec{s}$ represents a unit vector around which the rotation is performed. Due to the tangent, the rotation vector is indeterminate when the rotation angle equals ±pi radians or ±180 deg. Values can be negative or positive.

References

[1] Dai, J.S. "Euler-Rodrigues formula variations, quaternion conjugation and intrinsic connections." *Mechanism and Machine Theory*, 92, 144-152. Elsevier, 2015.

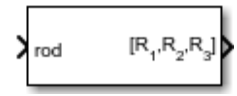
See Also

Direction Cosine Matrix to Rodrigues | Quaternions to Rodrigues | Rodrigues to Direction Cosine Matrix | Rodrigues to Rotation Angles | Rotation Angles to Rodrigues

Introduced in R2017a

Rodrigues to Rotation Angles

Convert Euler-Rodrigues vector to rotation angles
Library: Utilities / Axes Transformations



Description

The Rodrigues to Rotation Angles block converts the 3-element Euler-Rodrigues vector into rotation angles.

Ports

Input

`rod` — Euler-Rodrigues vector
3-element vector

Euler-Rodrigues vector determined from rotation angles.

Data Types: `double`

Output

`R1,R2,R3` — Rotation angles
3-element vector

Rotation angles, in radians, from which to determine the Euler-Rodrigues vector. Quaternion scalar is the first element.

Data Types: `double`

Parameters

`Rotation order` — Rotation order
ZYX (default) | ZYX | ZYZ | ZXY | ZXZ | YXZ | YXY | YZX | YZY | XYZ | XYX | XZY | XZX

Rotation order for three wind rotation angles.

For the 'ZYX', 'ZXY', 'YXZ', 'YZX', 'XYZ', and 'XZY' rotations, the block generates an R2 angle that lies between ±pi/2 radians (±90 degrees), and R1 and R3 angles that lie between ±pi radians (±180 degrees).

For the 'ZYZ', 'ZXZ', 'YXY', 'YZY', 'XYX', and 'XZX' rotations, the block generates an R2 angle that lies between 0 and pi radians (180 degrees), and R1 and R3 angles that lie between ±pi (±180 degrees). However, in the latter case, when R2 is 0, R3 is set to 0 radians.

Algorithms

An Euler-Rodrigues vector $\vec{b}$ represents a rotation by integrating a direction cosine of a rotation axis with the tangent of half the rotation angle as follows:

$$\vec{b} = \begin{bmatrix} b_x & b_y & b_z \end{bmatrix}$$

where:

$$b_x = \tan\left(\frac{1}{2}\theta\right)s_x,$$
$$b_y = \tan\left(\frac{1}{2}\theta\right)s_y,$$
$$b_z = \tan\left(\frac{1}{2}\theta\right)s_z$$

are the Rodrigues parameters. Vector $\vec{s}$ represents a unit vector around which the rotation is performed. Due to the tangent, the rotation vector is indeterminate when the rotation angle equals ±pi radians or ±180 deg. Values can be negative or positive.

References

[1] Dai, J.S. "Euler-Rodrigues formula variations, quaternion conjugation and intrinsic connections." *Mechanism and Machine Theory*, 92, 144-152. Elsevier, 2015.

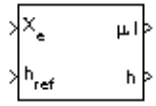
See Also

Direction Cosine Matrix to Rodrigues | Quaternions to Rodrigues | Rodrigues to Direction Cosine Matrix | Rodrigues to Quaternions | Rotation Angles to Rodrigues

Introduced in R2017a

Flat Earth to LLA

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 $(\bar{\iota})$, 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 $\left(\bar{f}\right)$ 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

$$d\mu = \text{atan}\left(\frac{1}{R_M}\right)dN$$
$$d\iota = \text{atan}\left(\frac{1}{R_N \cos\mu}\right)dE$$

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}).

$$h = -p_z - h_{ref}$$

Parameters

Units

Specifies the parameter and output units:

Units	Position	Equatorial Radius	Altitude
`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.

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

Introduced before R2006a

FlightGear Preconfigured 6DoF Animation

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 (μ), altitude (h), roll (ϕ), pitch (θ), and yaw (ψ), 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. If you generate code for your model using Simulink Coder and connect to the running target code using external mode simulation, 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” (Simulink).

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

Parameters

FlightGear version

> **Note** If you are using a FlightGear version older than 2.0, the model displays a notification from the Simulink Upgrade Advisor. Consider using the Upgrade Advisor to upgrade your FlightGear version. For more information, see “Supported FlightGear Versions” on page 2-19.

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

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

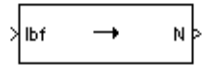
Send net_fdm Packet to FlightGear

Simulation Pace

Introduced before R2006a

Force Conversion

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 unit** and the **Final unit** lists.

Parameters

Initial unit

Specifies the input units.

Final unit

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

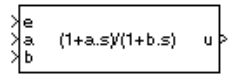
Temperature Conversion

Velocity Conversion

Introduced before R2006a

Gain Scheduled Lead-Lag

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.

Parameters

Initial state, x_initial

The initial internal state for the filter `x_initial`. Given this initial state, the initial output is given by

$$u\big|_{t=0} = \frac{x_initial + ae}{b}$$

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.

Introduced before R2006a

Generate Run Script

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 Parameters 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 2-31.

Parameters

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.

- `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.

Install FlightGear scenery during simulation (requires Internet connection)

Select this check box to direct FlightGear to automatically install required scenery while the simulator is running. Selecting this check box requires a steady Internet connection.

Disable FlightGear shader options

Select this check box to disable FlightGear shader options. Your computer built-in video card, such as NVIDIA cards, can conflict with FlightGear shaders. Consider selecting this check box if you have this conflict.

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.

Note The run script file name must be composed of ASCII characters.

Use these file extensions:

Platform	Extension
Windows	`.bat`
Linux and macOS	`.sh`

FlightGear base directory

Specify the name of your FlightGear installation folder.

Note FlightGear must be installed in a folder path name composed of ASCII characters.

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 `asbhl20` example.

See Also

FlightGear Preconfigured 6DoF Animation

Pack net_fdm Packet for FlightGear

Send net_fdm Packet to FlightGear

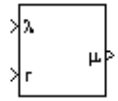
Receive net_ctrl Packet from FlightGear

Unpack net_ctrl Packet from FlightGear

Introduced before R2006a

Geocentric to Geodetic Latitude

Convert geocentric latitude to geodetic latitude

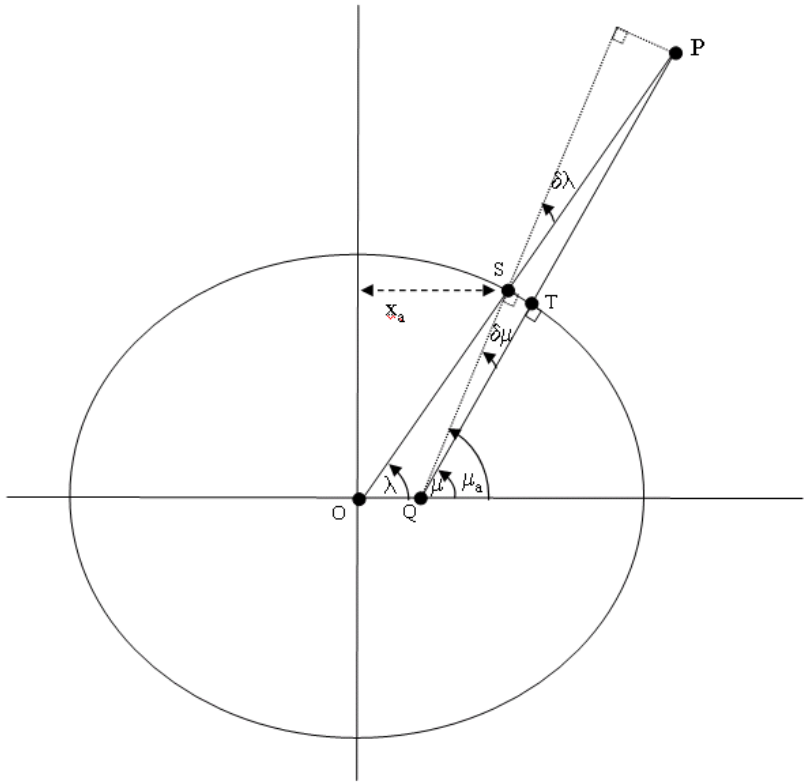


Library

Utilities/Axes Transformations

Description

The Geocentric to Geodetic Latitude block converts a geocentric latitude (λ) into geodetic latitude (μ). 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.



.

Given geocentric latitude (λ) 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 μ_a, the equation for an ellipse with equatorial radius (*R*), polar radius $\left((1-f)R\right)$ 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.

$$\mu_a = \operatorname{atan}\left(\frac{\tan\lambda}{(1-f)^2}\right)$$

Using the relationship $\tan\lambda = y_a / x_a$ and the two equations above, the resulting equation for μ_a is obtained.

$$\mu_a = \operatorname{atan}\left(\frac{\sqrt{R^2 - {x_a}^2}}{(1-f)x_a}\right)$$

The correct sign of μ_a is determined by testing λ and if λ is less than zero μ_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 $\left(r_a\right)$ 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 (ρ_a) at μ_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 ρ_a, the angular difference between geodetic latitude at S (μ) and geodetic latitude at P (μ_a) is defined as:

$$\delta\mu = \text{atan}\left(\frac{l\sin\delta\lambda}{\rho_a + h}\right)$$

Subtracting $\delta\mu$ from μ_a then gives μ.

$$\mu = \mu_a - \delta\mu$$

Parameters

Units

Specifies the parameter and output units:

Units	Radius from CG to Center of Planet	Equatorial Radius
`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`.

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. Latitude values can be any value. However, values of +90 and -90 may return unexpected values because of singularity at the poles.
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 ±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.edu/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

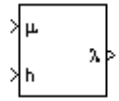
Geodetic to Geocentric Latitude

LLA to ECEF Position

Introduced before R2006a

Geodetic to Geocentric Latitude

Convert geodetic latitude to geocentric latitude



Library

Utilities/Axes Transformations

Description

The Geodetic to Geocentric Latitude block converts a geodetic latitude (μ) into geocentric latitude (λ). Geocentric latitude at the planet surface $\left(\lambda_s\right)$ 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 $\left(r_s\right)$ and geocentric latitude at the planet surface in the following relationship.

$$\lambda = \text{atan}\left(\frac{h\sin\mu + r_s \sin\lambda_s}{h\cos\mu + r_s\cos\lambda_s}\right)$$

Parameters

Units

Specifies the parameter and output units:

Units	Altitude	Equatorial Radius
`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`.

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 geodetic latitude, in degrees. Latitude values can be any value. However, values of +90 and -90 may return unexpected values because of singularity at the poles.
Second	Scalar	Contains the mean sea-level altitude (MSL).

Output	Dimension Type	Description
First	Scalar	Contains the geocentric latitude, in degrees.

Assumptions and Limitations

This implementation generates a geocentric latitude that lies between ±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

Introduced before R2006a

Heading Indicator

Display measurements for aircraft heading

Library

Flight Instruments

Description

The Heading Indicator block displays measurements for aircraft heading in degrees.

The block represents values between 0 and 360 degrees.

By default, the block displays the signal name at the top of the block.

Parameters

Connection

Select a signal to connect and display.

To view the data from a signal, select a signal in the model. The signal appears in the **Connection** table. Select the option button next to the signal you want to display. Click **Apply** to connect the signal.

The table has a row for the signal connected to the block. If there are no signals selected in the model, or the block is not connected to any signals, the table is empty.

Label

Name of connected signal.

- `Top`

  Show label at the top of the block.

- `Bottom`

  Show label at the bottom of the block.

- `Hide`

  Do not show the label or instructional text when the block is not connected.

See Also

Airspeed Indicator | Altimeter | Artificial Horizon | Climb Rate Indicator | Exhaust Gas Temperature (EGT) Indicator | Revolutions Per Minute (RPM) Indicator | Turn Coordinator

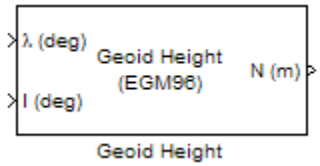
Topics

“Display Measurements with Cockpit Instruments” on page 2-47
“Flight Instrument Gauges” on page 2-46

Introduced in R2016a

Geoid Height

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.

Parameters

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. **Note** This block requires that you download geoid data for the EGM2008 Geopotential Model with the Add-On Explorer. Click the **Get data** button to start the Add-On Explorer. For more information, see `aeroDataPackage`. If the data is installed, the **Get data** button does not appear.
`Custom`	Custom geopotential model that you define in **Geopotential mat-file**. This block calculates geoid heights to an accuracy of 0.01 m for custom models. Selecting `Custom` enables the **Geopotential mat-file** parameter.

Geopotential mat-file

Specifies the MAT-file the defines your custom geopotential model. Selecting **Geopotential model** > **Custom** enables this parameter.

Data type

Specifies the data type of the input and output signals. From the list, select `double` or `single`.

Action for out-of-range input

Defines action for out-of-range input. Specify one:

`'Error'`
`'Warning'` (default)
`'None'`

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`

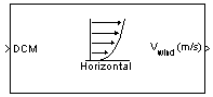
See Also

WGS84 Gravity Model, Spherical Harmonic Gravity Model

Introduced in R2010b

Horizontal Wind Model

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.

Parameters

Units

Specifies the input and output units:

Units	Wind Speed	Wind Velocity
`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

Horizontal Wind Model 07

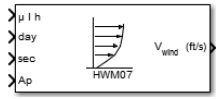
Von Karman Wind Turbulence Model (Continuous)

Wind Shear Model

Introduced before R2006a

Horizontal Wind Model 07

Implement Horizontal Wind Model 07



Library

Environment/Wind

Description

The Horizontal Wind Model 07 block implements the U.S. Naval Research Laboratory HWM™ routine to calculate the meridional and zonal components of the wind for a set of geophysical data: latitude, longitude, and altitude.

Parameters

Units

Specify the input and output units. The units you select determine the input and output wind speed and velocity, as shown in the table.

Units	Wind Speed	Wind Velocity
`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

Model

Select the horizontal wind model type for which to calculate the wind components.

- `Disturbance`

  Calculate the effect of only magnetic disturbances in the wind. For this model type, input Ap index values greater than or equal to 0.

- `Quiet`

  Calculate the horizontal wind model without the magnetic disturbances. For this model type, do not input an Ap index value.

- `Total`

  Calculate the combined effect of the quiet and magnetic disturbances. For this model type, input Ap index values greater than or equal to 0.

Action for out-of-range input

Specify the block behavior when the block inputs are out of range.

Value	Description
`None`	No action. The block imposes upper and lower limits on an input signal.
`Warning`	Warning in the Diagnostic Viewer, model simulation continues. For Accelerator and Rapid Accelerator modes, setting the action to Warning has no effect and the model behaves as though the action is set to `None`.
`Error` (default)	MATLAB returns an exception, model simulation stops. For Accelerator and Rapid Accelerator modes, setting the action to Error has no effect and the model behaves as though the action is set to `None`.

Inputs and Outputs

Input	Dimension Type	Description
First	Three-element vector of doubles	The input specifies the geodetic latitude (μ), longitude (l), and geopotential altitude (h) where the block implements the model. Latitude and longitude values are in degrees. The altitude value is in the units you selected in the **Units** parameter. Specify the altitude element as a value between 0 and 500 km.
Second	Scalar double	The input specifies the day of year in Universal Coordinated Time (UTC). The input specifies the day as a value between 1 and 366 (for a leap year).
Third	Scalar double	Contains elapsed seconds since midnight for the selected day, in UTC.
Fourth (Optional)	Scalar double	Contains the Ap index for the Universal Time (UT) when the block evaluates the model. Select the index from the NOAA National Geophysical Data Center, which contains 3 hour interval geomagnetic disturbance index values. If the Ap index value is greater than zero, the software takes into account magnetic effects during model evaluation. This input appears when you select the total or disturbance wind model type. Specify the Ap index as a value between 0 and 400. This input disappears when you select the quiet model type in the **Model** parameter.

Output	Dimension Type	Description
First	1-by-2 vector of doubles	The wind velocity vector contains the meridional and zonal wind components in that order.

Limitation

For code generation, use this block only for targets whose type is int 32 or higher.

See Also

Discrete Wind Gust Model | Dryden Wind Turbulence Model (Discrete) | Dryden Wind Turbulence Model (Continuous) | Horizontal Wind Model | Von Karman Wind Turbulence Model (Continuous) | Wind Shear Model

External Websites

NOAA National Geophysical Data Center

Introduced in R2014b

Horizontal Wind Model 14

Implement Horizontal Wind Model 14
Library: Environment / Wind

Description

The Horizontal Wind Model 14 block implements the U.S. Naval Research Laboratory (HWM) routine to calculate the meridional and zonal components of the wind for a set of geophysical data: latitude, longitude, and altitude.

Limitation

For code generation, use this block only for targets whose type is int 32 or higher.

Ports

Input

`First` — geodetic latitude (μ), longitude (`l`), and geopotential altitude (`h`)
three-element vector of doubles

The input specifies the geodetic latitude (μ), longitude (`l`), and geopotential altitude (`h`) where the block implements the model.

Latitude and longitude values are in degrees.

The altitude value is in the units you selected in the **Units** parameter. Specify the altitude element as a value between 0 and 500 km.

Second — day of year
scalar double

The input specifies the day of year in Universal Coordinated Time (UTC). The input specifies the day as a value between 1 and 366 (for a leap year).

`Third` — elapsed seconds
scalar double

Contains elapsed seconds since midnight for the selected day, in UTC.

`Fourth (Optional)` — Ap index
scalar double

Contains the Ap index for the Universal Time (UT) when the block evaluates the model. Select the index from the NOAA National Geophysical Data Center, which contains 3 hour interval geomagnetic disturbance index values. If the Ap index value is greater than zero, the software takes into account magnetic effects during model evaluation.

Output

`First` — wind velocity vector
1-by-2 vector of doubles

The wind velocity vector contains the meridional and zonal wind components in that order.

Parameters

`Units` — input and output units
`Metric (MKS)` (default) | `English (Velocity in ft/s)` | `English (Velocity in kts)`

Specify the input and output units. The units you select determine the input and output wind speed and velocity, as shown in the table.

Units	Wind Speed	Wind Velocity
`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

`Model` — horizontal wind model
`Quiet` (default) | `Total` | `Disturbance`

Select the horizontal wind model type for which to calculate the wind components.

- `Quiet`

  Calculate the horizontal wind model without the magnetic disturbances. For this model type, do not input an Ap index value.

- `Total`

  Calculate the combined effect of the quiet and magnetic disturbances. For this model type, input Ap index values greater than or equal to zero.

- `Disturbance`

  Calculate the effect of magnetic disturbances in the wind. For this model type, input Ap index values greater than or equal to zero.

`Action for out-of-range input` — block behavior
`Error` (default) | `Warning` | `None`

Specify the block behavior when the block inputs are out of range.

Value	Description
`Error` (default)	MATLAB returns an exception, and model simulation stops. For Accelerator and Rapid Accelerator modes, setting the action to `Error` has no effect and the model behaves as though the action is set to `None`.
`Warning`	Warning in the Diagnostic Viewer, and model simulation continues. For Accelerator and Rapid Accelerator modes, setting the action to `Warning` has no effect and the model behaves as though the action is set to `None`.
`None`	No action. The block imposes upper and lower limits on an input signal.

See Also

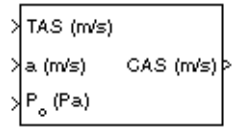
External Websites

NOAA National Geophysical Data Center

Introduced in R2016b

Ideal Airspeed Correction

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 (δ), 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.

Parameters

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

Airspeed input

Specify the airspeed input type:

`TAS`	True airspeed
`EAS`	Equivalent airspeed
`CAS`	Calibrated airspeed

Airspeed output

Specify the airspeed output type:

Velocity Input	Velocity Output
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

Input	Dimension Type	Description
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, γ.

Examples

See the `aeroblk_indicated` model and the `aeroblk_calibrated` model for examples of this block.

References

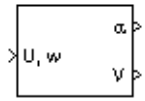
Lowry, J. T., *Performance of Light Aircraft*, AIAA Education Series, Washington, DC, 1999.

Aeronautical Vestpocket Handbook, United Technologies Pratt & Whitney, August, 1986.

Introduced before R2006a

Incidence & Airspeed

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.

$$\alpha = \mathrm{atan}\left(\frac{w}{u}\right)$$
$$V = \sqrt{u^2 + w^2}$$

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.

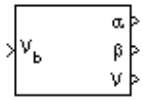
See Also

Incidence, Sideslip & Airspeed

Introduced before R2006a

Incidence, Sideslip & Airspeed

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.

$$\alpha = a\tan\left(\frac{w}{u}\right)$$

$$\beta = a\sin\left(\frac{v}{V}\right)$$

$$V = \sqrt{u^2 + v^2 + w^2}$$

$$\alpha = \text{atan}\left(\frac{w}{u}\right)$$

$$\beta = \text{asin}\left(\frac{v}{V}\right)$$

$$V = \sqrt{u^2 + v^2 + w^2}$$

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.

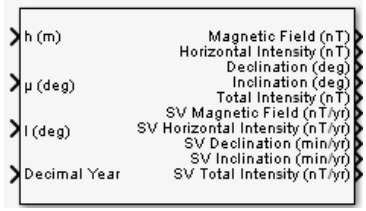
See Also

Incidence & Airspeed

Introduced before R2006a

International Geomagnetic Reference Field 12

Calculate Earth magnetic field and secular variation using 12th generation International Geomagnetic Reference Field



Library

Environment/Gravity

Description

The International Geomagnetic Reference Field 12 block calculates the Earth magnetic field and secular variation using the 12th generation International Geomagnetic Reference Field. It calculates these values at a location and time that you define.

Parameters

Units

Specifies the parameter and output units.

Units	Height
`Metric (MKS)`	Meters
`English`	Feet

Input decimal year

When you select this check box, the decimal year is an input for the International Geomagnetic Reference Field 12 block. Otherwise, specify a date using the **Month**, **Day**, and **Year** parameters.

Month

Specifies the month to calculate decimal year.

Day

Specifies the day to calculate decimal year.

Year

Specifies the year to calculate decimal year. From the list, select from `1900` to `2020`.

Action for out-of-range input

Specifies whether out-of-range input causes 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). *Z* 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)
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)

Secular Variance	Description
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.
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. This code shows how to calculate the decimal year, `dyear`, for March 21, 2015: `dyear = decyear('21-March-2015','dd-mmm-yyyy')`

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.

Output	Dimension Type	Description
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.
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 600,000 m.

This block is valid between the years 1900 and 2020.

This site shows additional limitations:

`http://www.ngdc.noaa.gov/IAGA/vmod/igrfhw.html`

References

International Association of Geomagnetism and Aeronomy. 12th 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.

Lowes, F. J. “The International Geomagnetic Reference Field: A 'Health' Warning.” January, 2010. http://www.ngdc.noaa.gov/IAGA/vmod/igrfhw.html.

Introduced in R2015b

Interpolate Matrix(x)

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

$x = [\ x_1\ x_2\ x_3\ \ldots\ x_i\ x_{i+1}\ \ldots\ x_n\ ]$.

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;
```

Parameters

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 λ.

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

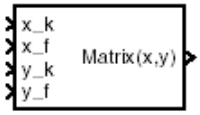
Interpolate Matrix(x,y)

Interpolate Matrix(x,y,z)

Introduced before R2006a

Interpolate Matrix(x,y)

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, $\boldsymbol{x} = [\ x_1\ x_2\ x_3 \ldots\ x_i\ x_{i+1}\ \ldots\ x_n]$ and $\boldsymbol{y} = [\ y_1\ y_2\ y_3\ \ldots\ y_j\ y_{j+1}\ \ldots\ y_m]$. 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;
```

Parameters

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 λ_x.
Third		Contains the second interpolation index *j*.
Fourth		Contains the second interpolation fraction λ_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

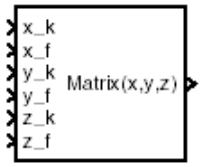
Interpolate Matrix(x)

Interpolate Matrix(x,y,z)

Introduced before R2006a

Interpolate Matrix(x,y,z)

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:

$x = [x_1\ x_2\ x_3\ \ldots\ x_i\ x_{i+1}\ \ldots\ x_n]$

$y = [y_1\ y_2\ y_3\ \ldots\ y_j\ y_{j+1}\ \ldots\ y_m]$

$z = [z_1\ z_2\ z_3\ \ldots\ z_k\ z_{k+1}\ \ldots\ z_p]$

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

$$
\begin{aligned}
&(1-\lambda_z)\Big\{(1-\lambda_y)\Big[(1-\lambda_x)M(x_i,y_j,z_k)+\lambda_x M(x_{i+1},y_j,z_k)\Big]\\
&\qquad+\lambda_y\Big[(1-\lambda_x)M(x_i,y_{j+1},z_k)+\lambda_x M(x_{i+1},y_{j+1},z_k)\Big]\Big\}\\
&+\lambda_z\Big\{(1-\lambda_y)\Big[(1-\lambda_x)M(x_i,y_j,z_{k+1})+\lambda_x M(x_{i+1},y_j,z_{k+1})\Big]\\
&\qquad+\lambda_y\Big[(1-\lambda_x)M(x_i,y_{j+1},z_{k+1})+\lambda_x M(x_{i+1},y_{j+1},z_{k+1})\Big]\Big\}
\end{aligned}
$$

where the three interpolation fractions are denoted by

$$
\begin{aligned}
\lambda_x &= (x-x_i)/(x_{i+1}-x_i)\\
i_y &= (y-y_j)/(y_{j+1}-y_j)\\
\lambda_z &= (z-z_k)/(z_{k+1}-z_k)
\end{aligned}
$$

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;`

Parameters

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 λ_x.
Third		Contains the second interpolation index j.
Fourth		Contains the second interpolation fraction λ_y.
Fifth		Contains the third interpolation index k.
Sixth		Contains the third interpolation fraction λ_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)

Introduced before R2006a

Invert 3x3 Matrix

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.

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

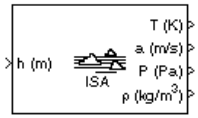
Create 3x3 Matrix

Determinant of 3x3 Matrix

Introduced before R2006a

ISA Atmosphere Model

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.

Parameters

Change atmospheric parameters

Select to customize various atmospheric parameters to be different from the ISA values.

Selecting this check box enables the parameters .

Acceleration due to gravity (m/s^2)

Enter acceleration from gravity in m/s^2.

Ratio of specific heats

Enter a ratio of specific hits.

Characteristic gas constant (J/Kg/K)

Enter the characteristic gas constant in J/Kg/K.

Lapse rate (K/m)

Enter the lapse rate in K/m.

Height of troposphere (m)

Enter the height of the troposphere in m.

Height of tropopause (m)

Enter the height of the tropopause in m.

Air density at mean sea level (Kg/m^3)

Enter the air density at mean sea level in Kg/m^3.

Ambient pressure at mean sea level (N/m^2)

Enter the ambient pressure at mean sea level in N/m^2.

Ambient temperature at mean sea level (K)

Enter the ambient temperature at mean sea level in K.

Lowest altitude (m)

Enter the lowest altitude in m.

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

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.

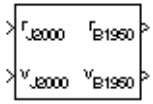
See Also

COESA Atmosphere Model, CIRA-86 Atmosphere Model, Lapse Rate Model

Introduced before R2006a

Julian Epoch to Besselian Epoch

Transform position and velocity components from Standard Julian Epoch (J2000) to discontinued Standard Besselian Epoch (B1950)



Library

Utilities/Axes Transformations

Description

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

$$\left(\bar{M}_{rr}, \bar{M}_{vr}, \bar{M}_{rv}, \bar{M}_{vv}\right)$$

are defined as:

$$\bar{M}_{rr} = \begin{bmatrix} 0.9999256782 & -0.0111820611 & -0.0048579477 \\ 0.0111820610 & 0.9999374784 & -0.0000271765 \\ 0.0048579479 & -0.0000271474 & 0.9999881997 \end{bmatrix}$$

$$\bar{M}_{vr} = \begin{bmatrix} 0.00000242395018 & -0.00000002710663 & -0.00000001177656 \\ 0.00000002710663 & 0.00000242397878 & -0.00000000006587 \\ 0.00000001177656 & -0.00000000006582 & 0.00000242410173 \end{bmatrix}$$

$$\bar{M}_{rv} = \begin{bmatrix} -0.000551 & -0.238565 & 0.435739 \\ 0.238514 & -0.002667 & -0.008541 \\ -0.435623 & 0.012254 & 0.002117 \end{bmatrix}$$

$$\bar{M}_{vv} = \begin{bmatrix} 0.99994704 & -0.01118251 & -0.00485767 \\ 0.01118251 & 0.99995883 & -0.00002718 \\ 0.00485767 & -0.00002714 & 1.00000956 \end{bmatrix}$$

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.

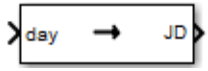
See Also

Besselian Epoch to Julian Epoch

Introduced before R2006a

Julian Date Conversion

Calculate Julian date or modified Julian date



Library

Utilities/Unit Conversions

Description

This block converts the specified date to the Julian date or modified Julian date.

Parameters

Year

Specify the year used to calculate the Julian date. Enter a double value that is a whole number greater than 1, such as `2013`.

Month

Specify the month used to calculate the Julian date. From the list, select the month from `January` to `December`.

Day

Specify the day used to calculate the Julian date. From the list, select the day from `1` to `31`.

Hour

Specify the hour used to calculate the Julian date. Enter a double value that is a whole number, from `0` to `24`.

Minutes

Specify the minutes used to calculate the Julian date. Enter a double value that is a whole number, from `0` to `60`.

Seconds

Specify the seconds used to calculate the Julian date. Enter a double value that is a whole number, from `0` to `60`.

Calculate modified Julian date

Select this check box to calculate the modified Julian date (MJD) for corresponding elements of the year, month, day, hour, minute, and second.

Time Increment

Specify the time increment between the specified date and the desired model simulation time. The block adjusts the calculated Julian date to take into account the time increment from model simulation. For example, selecting `Day` and connecting a simulation timer to the port means that each time increment unit is one day and the block adjusts its calculation based on that simulation time.

This parameter corresponds to the first block input, the clock source.

Possible values are `Day`, `Hour`, `Min`, `Sec`, and `None`. If you select `None`, the calculated Julian date does not take into account the model simulation time. Selecting this option removes the first block input.

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.

Inputs and Outputs

Input	Dimension Type	Description
First (Optional)	Scalar	Clock source for model simulation.

Output	Dimension Type	Description
First	Scalar	Julian date.

Assumptions and Limitations

This block is valid for all common era (CE) dates in the Gregorian calendar.

The calculation of Julian date does not take into account leap seconds.

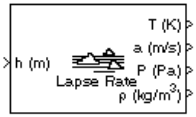
See Also

`juliandate`

Introduced in R2013b

Lapse Rate Model

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)
ρ_0	Air density at mean sea level in kg/m^3
P_0	Static pressure at mean sea level in N/m^2
h	Altitude in m
hts	Height of the troposphere in m
T	Absolute temperature at altitude h in kelvin (K)
ρ	Air density at altitude h in kg/m^3
P	Static pressure at altitude h in N/m^2
a	Speed of sound at altitude h in m/s^2
L	Lapse rate in K/m
R	Characteristic gas constant J/kg-K
γ	Specific heat ratio
g	Acceleration due to gravity in m/s^2

The Lapse Rate Model block icon displays the input and output metric units.

Parameters

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 γ.

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 (ρ_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.

Output	Dimension Type	Description
First		Contains the temperature.
Second		Contains the speed of sound.

Output	Dimension Type	Description
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

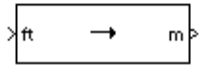
COESA Atmosphere Model

ISA Atmosphere Model

Introduced before R2006a

Length Conversion

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 unit** and the **Final unit** lists.

Parameters

Initial unit

Specifies the input units.

Final unit

Specifies the output units.

The following conversion units are available:

`m`	Meters
`ft`	Feet
`km`	Kilometers
`in`	Inches

`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

Introduced before R2006a

Linear Second-Order Actuator

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.

Parameters

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.

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.

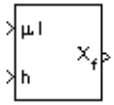
See Also

Nonlinear Second-Order Actuator

Introduced in R2012a

LLA to ECEF Position

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 $(\bar{p})$. Latitude and longitude values can be any value. However, latitude values of +90 and -90 may return unexpected values because of singularity at the poles. The ECEF position is calculated from geocentric latitude at mean sea-level (λ_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.

$$\lambda_s = \text{atan}((1-f)^2 \tan\mu)$$

$$r_s = \sqrt{\frac{R^2}{1+\left(1/(1-f)^2 - 1\right)\sin^2\lambda_s}}$$

Parameters

Units

Specifies the parameter and output units:

Units	Altitude	Equatorial Radius	Position
`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

Input	Dimension Type	Description
First	2-by-1 vector	Contains the geodetic latitude and longitude, in degrees.

Input	Dimension Type	Description
Second	Scalar	Contains the altitude above the planetary ellipsoid.

Output	Dimension Type	Description
First	3-by-1 vector	Contains the position in ECEF frame, in same units as altitude.

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 2-10.

Direction Cosine Matrix ECEF to NED

Direction Cosine Matrix ECEF to NED to Latitude and Longitude

ECEF Position to LLA

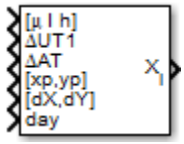
Flat Earth to LLA

Radius at Geocentric Latitude

Introduced before R2006a

LLA to ECI Position

Convert latitude, longitude, altitude (LLA) coordinates to Earth-centered inertial (ECI) coordinates



Library

Utilities/Axes Transformations

Description

LLA to ECI Position block converts latitude, longitude, altitude (LLA) coordinates to Earth-centered inertial (ECI) position coordinates, based on the specified reduction method and Universal Coordinated Time (UTC), for the specified time and geophysical data. Latitude and longitude values can be any value. However, latitude values of +90 and -90 may return unexpected values because of singularity at the poles.

Parameters

Reduction

Reduction method to convert the coordinates. Select one of the following:

- `IAU-76/FK5`

  Reduce the calculation using the International Astronomical Union (IAU)-76/Fifth Fundamental Catalogue (FK5) (IAU-76/FK5) reference system. Choose this reduction method if the reference coordinate system for the conversion is FK5.

  Note This method uses the IAU 1976 precession model and the IAU 1980 theory of nutation to reduce the calculation. This model and theory are no longer current,

but the software provides this reduction method for existing implementations. Because of the polar motion approximation that this reduction method uses, the block calculates the transformation matrix rather than the direction cosine matrix.

- `IAU-2000/2006`

  Reduce the calculation using the International Astronomical Union (IAU)-2000/2006 reference system. Choose this reduction method if the reference coordinate system for the conversion is IAU-2000. This reduction method uses the P03 precession model to reduce the calculation.

Year

Specify the year used to calculate the Universal Coordinated Time (UTC) date. Enter a double value that is a whole number greater than 1, such as `2013`.

Month

Specify the month used to calculate the UTC date. From the list, select the month from `January` to `December`.

Day

Specify the day used to calculate the UTC date. From the list, select the day from `1` to `31`.

Hour

Specify the hour used to calculate the UTC date. Enter a double value that is a whole number from `0` to `24`.

Minutes

Specify the minutes used to calculate the UTC date. Enter a double value that is a whole number from `0` to `60`.

Seconds

Specify the seconds used to calculate the UTC date. Enter a double value that is a whole number from `0` to `60`.

Time Increment

Specify the time increment between the specified date and the desired model simulation time. The block adjusts the calculated direction cosine matrix to take into account the time increment from model simulation. For example, selecting `Day` and connecting a simulation timer to the port means that each time increment unit is one day. The block adjusts its calculation based on that simulation time.

This parameter corresponds to the sixth block input, the clock source.

Possible values are `Day`, `Hour`, `Min`, `Sec`, and `None`. If you select `None`, the calculated Julian date does not take into account the model simulation time. Selecting this option removes the fifth block input.

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.

Higher accuracy parameters

Select this check box to enable the following inputs. These inputs let you better control the conversion result. See “Inputs and Outputs” on page 4-394 for a description.

[μ l h]
$\Delta UT1$
ΔAT
[xp,yp]
[$\Delta\delta\psi$, $\Delta\delta\varepsilon$] or [dX,dY]
day

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)`.

Earth model

Specifies the planet model to use: `Custom` or `WGS84`.

Flattening

Specifies the flattening of the planet. This option is available only with **Earth model** **Custom**.

Equatorial radius

Specifies the radius of the planet at its equator. This option is available only with **Earth model Custom**.

Inputs and Outputs

Input	Dimension Type	Description
First	Three-element vector	[μ l h], latitude, longitude, and altitude values of coordinates to convert, in degrees.
Second (Optional)	Scalar	$\Delta UT1$, difference between UTC and Universal Time (UT1), in seconds, for which the function calculates the direction cosine or transformation matrix, for example, `0.234`.
Third (Optional)	Scalar	ΔAT, difference between International Atomic Time (IAT) and UTC, in seconds, for which the function calculates the direction cosine or transformation matrix, for example, `32`.
Fourth (Optional)	1-by-2 array	[xp,yp], polar displacement of the Earth, in radians, from the motion of the Earth crust, along the x- and y-axes, for example, `[-0.0682e-5 0.1616e-5]`.

Input	Dimension Type	Description
Fifth (Optional)	1-by-2 array	• If reduction method is `IAU-2000/2006`, this input is the adjustment to the location of the Celestial Intermediate Pole (CIP), specified in radians. This location ([d*X*,d*Y*]) is along the *x*- and *y*-axes, for example, `[-0.2530e-6 -0.0188e-6]`. • If reduction method is `IAU-76/FK5`, this input is the adjustment to the longitude ([Δδψ, Δδε]), specified in radians, for example, `[-0.2530e-6 -0.0188e-6]`. For historical values, see the International Earth Rotation and Reference Systems Service website (`http://www.iers.org`) and navigate to the Earth Orientation Data Data/Products page.
Sixth	Scalar	Time increment, for example, the Clock block. If the **Higher accuracy parameters** check box is cleared and the **Time Increment** parameter is a value other than `None`, the block has no input.

Output	Dimension Type	Description
First	3-by-1 element vector	Original position vector with respect to the ECI reference system.

See Also

ECI Position to LLA

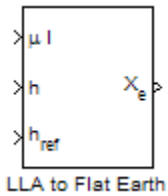
Topics

http://www.iers.org

Introduced in R2014a

LLA to Flat Earth

Estimate flat Earth position from geodetic latitude, longitude, and altitude



Library

Utilities/Axes Transformations

Description

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 $(\bar{p})$. Latitude and longitude values can be any value. However, latitude values of +90 and -90 may return unexpected values because of singularity at the poles. 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

$$dN = \frac{d\mu}{\operatorname{atan}\left(\frac{1}{R_M}\right)}$$

$$dE = \frac{d\imath}{\operatorname{atan}\left(\frac{1}{R_N \cos\mu_0}\right)}$$

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

(ψ)

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}).

$$p_z = -h - h_{ref}$$

Parameters

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.

Input	Dimension Type	Description
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.

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

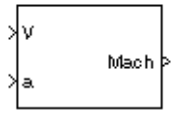
LLA to ECEF Position

Radius at Geocentric Latitude

Introduced in R2011a

Mach Number

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.

Inputs and Outputs

Input	Dimension Type	Description
First	*m*-by-3	Contains the velocity vector.
Second	1-by-1	Contains the speed of sound.

Output	Dimension Type	Description
First	*m*-by-1	Contains the Mach number.

Examples

See Airframe in the `aeroblk_HL20` model for an example of this block.

See Also

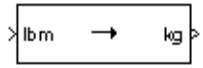
Aerodynamic Forces and Moments

Dynamic Pressure

Introduced before R2006a

Mass Conversion

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 unit** and the **Final unit** lists.

Parameters

Initial unit

Specifies the input units.

Final unit

Specifies the output units.

The following conversion units are available:

`lbm`	Pound mass
`kg`	Kilograms
`slug`	Slugs

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

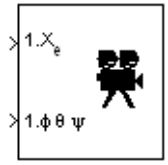
Temperature Conversion

Velocity Conversion

Introduced before R2006a

MATLAB Animation

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 Parameters 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.

Parameters

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}`
- Character vector with single quotes, for example, `'astredwedge.mat'`
- Mixed cell array of variable names and character vectors, 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.

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

This block has the following inputs:

Input	Dimension Type	Description
First	Vector	Contains the downrange position, the crossrange position, and the altitude of the vehicle in Earth coordinates. The number indicates the vehicle number.
Second	Vector	Contains the Euler angles (roll, pitch, and yaw) of the vehicle. The number indicates the vehicle number.

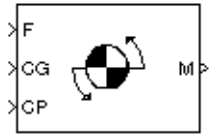
See Also

`Aero.Animation` in the Aerospace Toolbox documentation

Introduced in R2007a

Moments About CG Due to Forces

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.

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.

See Also

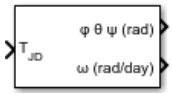
Aerodynamic Forces and Moments

Estimate Center of Gravity

Introduced before R2006a

Moon Libration

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 Julian date input for the block, you can use the Julian Date Conversion block or the Aerospace Toolbox `juliandate` function to calculate the Julian date. You can also calculate your own Julian date and input it using the Constant block.

Parameters

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`

  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	Released in 1998. This ephemerides takes into account the Julian date range 2305424.50 (December 9, 1599) to 2525008.50 (February 20, 2201). This block implements these ephemerides with respect to the International Celestial Reference Frame version 1.0, adopted in 1998.
DE421	Released in 2008. This ephemerides takes into account the Julian date range 2414992.5 (December 4, 1899) to 2469808.5 (January 2, 2050). This block implements these ephemerides with respect to the International Celestial Reference Frame version 1.0, adopted in 1998.
DE423	Released in 2010. This ephemerides takes into account the Julian date range 2378480.5 (December 16, 1799) to 2524624.5 (February 1, 2200). This block implements these ephemerides with respect to the International Celestial Reference Frame version 2.0, adopted in 2010.

Ephemerides Model	Description
`DE430`	Released in 2013. This ephemerides takes into account the Julian date range 2287184.5 (December 21, 1549) to 2688976.5 (January 25, 2650). This block implements these ephemerides with respect to the International Celestial Reference Frame version 2.0, adopted in 2010.
`DE432t`	Released in April 2014. This ephemerides takes into account the Julian date range 2287184.5, (December 21, 1549) to 2688976.5, (January 25, 2650). This block implements these ephemerides with respect to the International Celestial Reference Frame version 2.0, adopted in 2010.

Note This block requires that you download ephemeris data with the Add-On Explorer. Click the **Get data** button to start the Add-On Explorer. For more information, see `aeroDataPackage`. If the data is installed, the **Get data** button does not appear.

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	• Julian date (T_{JD}) (default) — One input. Specify a date between the minimum and maximum Julian date. • Fixed Julian date ($T0_{JD}$) plus the elapsed Julian time (ΔT_{JD}) between the fixed date and the ephemeris time. — Two inputs, where the first input is $T0_{JD}$ and the second input is ΔT_{JD}. ΔT_{JD} must be a positive number. The sum of $T0_{JD}$ and ΔT_{JD} must fall between the minimum and maximum Julian date. The block **Epoch** parameter controls the number of block inputs. See the **Ephemerides** parameter for the minimum and maximum Julian dates.
Second (Optional)	Scalar	ΔT_{JD} — Elapsed Julian time (ΔT_{JD}) between the fixed date and the ephemeris time. The sum of $T0_{JD}$ and ΔT_{JD} must fall between the minimum and maximum Julian date. See the **Ephemerides** parameter for the minimum and maximum Julian date.

Output	Dimension Type	Description
First	Vector	Euler angles (φ θ ψ) for Moon attitude. Units are radians.
Second (Optional)	Vector	Moon libration Euler angular rates (ω). 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

Direction Cosine Matrix ECI to ECEF | Earth Nutation | Julian Date Conversion | Planetary Ephemeris

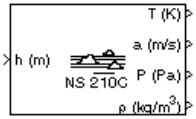
External Websites

`http://ssd.jpl.nasa.gov/?planet_eph_export`
`http://syrte.obspm.fr/jsr/journees2010/powerpoint/folkner.pdf`

Introduced in R2013a

Non-Standard Day 210C

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.

Parameters

Units

Specifies the input and output units:

Units	Height	Temperature	Speed of Sound	Air Pressure	Air Density
`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

Units	Height	Temperature	Speed of Sound	Air Pressure	Air Density
`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.

`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 **Atmospheric model type**
`Low pressure`	This option is available only when `Envelope` is selected for **Atmospheric model type**

Frequency of occurrence

Select percent of time the values would occur.

`Extreme values`	This option is available only when `Envelope` is selected for **Atmospheric model type**.
`1%`	Option always available
`5%`	This option is available only when `Envelope` is selected for **Atmospheric model type**.
`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-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

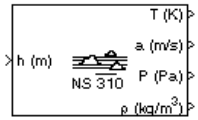
ISA Atmosphere Model

Non-Standard Day 310

Introduced before R2006a

Non-Standard Day 310

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.

Parameters

Units

Specifies the input and output units:

Units	Height	Temperature	Speed of Sound	Air Pressure	Air Density
`Metric (MKS)`	Meters	Kelvin	Meters per second	Pascal	Kilograms per cubic meter

Units	Height	Temperature	Speed of Sound	Air Pressure	Air Density
`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 **Atmospheric model type**.
`Low pressure`	This option is available only when `Envelope` is selected for **Atmospheric model type**.

Frequency of occurrence

Select percent of time the values would occur.

`Extreme values`	This option is available only when `Envelope` is selected for **Atmospheric model type**.
`1%`	Option always available
`5%`	This option is available only when `Envelope` is selected for **Atmospheric model type**.
`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.

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

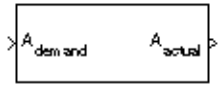
ISA Atmosphere Model

Non-Standard Day 210C

Introduced before R2006a

Nonlinear Second-Order Actuator

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.

Parameters

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.

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.

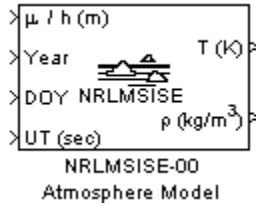
See Also

Linear Second-Order Actuator

Introduced in R2012a

NRLMSISE-00 Atmosphere Model

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) of the MSIS® class model. 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.

This block has the limitations of the NRLMSISE-00 model. For more information, see the NRLMSISE-00 model documentation.

Note This block is valid only for altitudes between 0 and 1,000,000 meters (1,000 kilometers).

Parameters

Units

Specifies the input and output units:

Units	Temperature	Height	Density
`Metric (MKS)`	Kelvin	Meters	kg/m^3, some density outputs $1/m^3$
`English`	Rankine	Feet	lbm/ft^3, some density outputs $1/ft^3$

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
`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)
`Flags(22)`	All lower mesosphere temperature (TN3) variations

Field	Description
`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 geodetic 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.

Input	Dimension Type	Description
Eight (Optional)	*N*-by-7 array	Contains an array of *N*-by-7 of magnetic index information. If you specify *magneticIndex*, you must also specify *f107Average* and *f107Daily*. This information consists of: Daily magnetic index (AP) 3 hour AP for current time 3 hour AP for 3 hours before current time 3 hour AP for 6 hours before current time 3 hour AP for 9 hours before current time Average of eight 3 hour AP indices from 12 to 33 hours before current time Average of eight 3 hour AP indices from 36 to 57 hours before current time The effects of daily magnetic index are not large or established below 80,000 m. As a result, the function sets the default value to 4. See the limitations in Description on page 4-428 for more 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.

Field	Description
`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)`

Field	Description
`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)
`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

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/GEOMAGNETIC_DATA/INDICES/KP_AP/`
- `ftp://ftp.ngdc.noaa.gov/STP/space-weather/solar-data/solar-features/solar-radio/noontime-flux/penticton/`

The format for the data indices for these values are located here:

`ftp://ftp.ngdc.noaa.gov/STP/GEOMAGNETIC_DATA/INDICES/KP_AP/kp_ap.fmt`

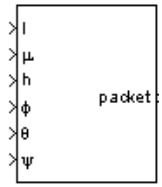
References

`http://ccmc.gsfc.nasa.gov/modelweb/atmos/nrlmsise00.html`

Introduced in R2007b

Pack net_fdm Packet for FlightGear

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.

Parameters

FlightGear version

Select your FlightGear software version: `v2.0`, `v2.4`, `v2.6`, `v2.8`, `v2.10`, `v2.12`, `v3.0`, `v3.2`, `v3.4`, `v2016.1`, `v2016.3`, `v2017.1`, `v2017.3`.

Note If you are using a FlightGear version older than 2.0, the model displays a notification from the Simulink Upgrade Advisor. Consider using the Upgrade Advisor to upgrade your FlightGear version. For more information, see “Supported FlightGear Versions” on page 2-19.

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.

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

Output Signal Supported for FlightGear 2.0/2.4/2.6/2.8/2.10/2.12/3.0/3.2/3.4/2016.1/2016.3/2017.1/2017.3

Output	Dimension Type	Description
First	Array	Contains the packet generated for FlightGear.

Input Signals Supported for FlightGear 2.0/2.4/2.6/2.8/2.10/2.12/3.0/3.2/3.4/2016.1/2016.3/2017.1/2017.3

This table lists all the input signals supported for Versions 2.0, 2.4, 2.6, 2.8, 2.10, 2.12, 3.0, 3.2, 3.4, 2016.1, 2016.3, 2017.1, 2017.3:

Signal Group 1: Position/altitude inputs

Name	Units	Type	Width	Description
longitude	rad	double	1	Geodetic longitude.
latitude	rad	double	1	Geodetic latitude.
altitude	m	double	1	Altitude above sea level.
theta	rad	single	1	Pitch.
phi	rad	single	1	Roll.
psi	rad	single	1	Yaw.

Signal Group 2: Velocity/acceleration inputs

Name	Units	Type	Width	Description
alpha	rad	single	1	Angle of Attack.
beta	rad	single	1	Sideslip angle.
dphi/dt	rad/sec	single	1	Roll rate.
dtheta/dt	rad/sec	single	1	Pitch rate.
dpsi/dt	rad/sec	single	1	Yaw rate.
Vcas	knot	single	1	Calibrated airspeed.
climb_rate	feet/sec	single	1	Rate of climb.
v_north	feet/sec	single	1	North velocity in body frame.
v_east	feet/sec	single	1	East velocity in body frame.
v_down	feet/sec	single	1	Down velocity.
v_wind_body_north	feet/sec	single	1	North velocity in body frame relative to local airmass.
v_wind_body_east	feet/sec	single	1	East velocity in body frame relative to local airmass.
v_wind_body_down	feet/sec	single	1	Down velocity in body frame relative to airmass.
Axpilot	feet/sec^2	single	1	*X* acceleration in body frame.
Aypilot	feet/sec^2	single	1	*Y* acceleration in body frame.
Azpilot	feet/sec^2	single	1	*Z* acceleration in body frame.
stall_warning	—	single	1	Amount of stall [0-1].
slip_deg	deb	single	1	Slip ball deflection.

Signal Group 3: Control surface position inputs

Name	Units	Type	Width	Description
elevator	1 (dimensionless)	single	1	Normalized elevator position.
elevator_trim_tab	1 (dimensionless)	single	1	Normalized elevator trim tab position.
left_flap	1 (dimensionless)	single	1	Normalized left flap position.
right_flap	1 (dimensionless)	single	1	Normalized right flap position.
left_aileron	1 (dimensionless)	single	1	Normalized left aileron position.
right_aileron	1 (dimensionless)	single	1	Normalized right aileron position
rudder	1 (dimensionless)	single	1	Normalized rudder position.
nose_wheel	1 (dimensionless)	single	1	Normalized nose wheel position.
speedbrake	1 (dimensionless)	single	1	Normalized speedbrake position.
spoilers	1 (dimensionless)	single	1	Normalized spoilers position.

Signal Group 4: Engine/fuel inputs

Name	Units	Type	Width	Description
num_engines	—	uint32	1	Number of engines.
eng_state	—	uint32	4	Engine state (off, cranking, running).
rpm	rev/min	single	4	Engine RPM.
fuel_flow	gal/hour	single	4	Fuel flow.
fuel_px	psi	single	4	Fuel pressure.
egt	deg F	single	4	Exhaust gas temperature.
cht	deg F	single	4	Cylinder head temperature.
mp_osi	psi	single	4	Manifold pressure.
tit	deg F	single	4	Turbine inlet temperature.
oil_temp	deg F	single	4	Oil temperature.
oil_px	psi	single	4	Oil pressure.
num_tanks	—	uint32	1	Number of fuel tanks.
fuel_quantity	gal	single	4	Fuel quantity per tank.

Signal Group 5: Landing gear inputs

Name	Units	Type	Width	Description
num_wheels	—	uint32	1	Number of wheels.
wow	—	uint32	3	Weight on wheels switch.
gear_pos	—	single	3	Landing gear normalized position.
gear_steer	—	single	3	Landing gear normalized steering.
gear_compression	—	single	3	Landing gear normalized compression.

Signal Group 6: Environment inputs

Name	Units	Type	Width	Description
agl	m	single	1	Above ground level.
cur_time	sec	uint32	1	Current UNIX® time.
warp	sec	uint32	1	Offset in seconds to UNIX time.
visibility	m	single	1	Visibility in meters (for visual effects).

Examples

See `asbhl20` 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

Introduced before R2006a

Pilot Joystick

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.

Parameters

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.

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
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

Introduced before R2006a

Pilot Joystick All

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.

Parameters

Joystick ID

Specify the joystick ID: `Joystick 1`, `Joystick 2`, or `None`.

Output configuration

Specify the output configuration: `FourAxis` (see Pilot Joystick) or `AllOutputs`. `AllOutputs` is the default.

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
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.

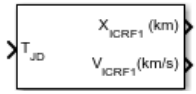
See Also

Pilot Joystick, Simulation Pace

Introduced in R2007a

Planetary Ephemeris

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 Julian date input for the block, you can use the Julian Date Conversion block or the Aerospace Toolbox `juliandate` function to calculate the Julian date. You can also calculate your own Julian date and input it using the Constant block.

This block implements the position and velocity using the International Celestial Reference Frame. If you want the planetary ephemeris position value relative to Earth in Earth-fixed (ECEF) coordinates, use the Direction Cosine Matrix ECI to ECEF block.

Parameters

Units

Specify the output units:

Units	Position	Velocity
`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.

- `T0 and elapsed Julian time`

  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`	Released in 1998. This ephemerides takes into account the Julian date range 2305424.50 (December 9, 1599) to 2525008.50 (February 20, 2201). This block implements these ephemerides with respect to the International Celestial Reference Frame version 1.0, adopted in 1998.

Ephemerides Model	Description
DE421	Released in 2008. This ephemerides takes into account the Julian date range 2414992.5 (December 4, 1899) to 2469808.5 (January 2, 2050). This block implements these ephemerides with respect to the International Celestial Reference Frame version 1.0, adopted in 1998.
DE423	Released in 2010. This ephemerides takes into account the Julian date range 2378480.5 (December 16, 1799) to 2524624.5 (February 1, 2200). This block implements these ephemerides with respect to the International Celestial Reference Frame version 2.0, adopted in 2010.
DE430	Released in 2013. This ephemerides takes into account the Julian date range 2287184.5 (December 21, 1549) to 2688976.5 (January 25, 2650). This block implements these ephemerides with respect to the International Celestial Reference Frame version 2.0, adopted in 2010.
DE432t	Released in April 2014. This ephemerides takes into account the Julian date range 2287184.5, (December 21, 1549) to 2688976.5, (January 25, 2650). This block implements these ephemerides with respect to the International Celestial Reference Frame version 2.0, adopted in 2010.

Note This block requires that you download ephemeris data with the Add-On Explorer. Click the **Get data** button to start the Add-On Explorer. For more information, see `aeroDataPackage`. If the data is installed, the **Get data** button does not appear.

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	• Julian date (T_{JD}) (default) — One input. Specify a date between the minimum and maximum Julian date. • Fixed Julian date ($T0_{JD}$) plus the elapsed Julian time (ΔT_{JD}) between the fixed date and the ephemeris time. — Two inputs, where the first input is $T0_{JD}$ and the second input is ΔT_{JD}. ΔT_{JD} must be a positive number. The sum of $T0_{JD}$ and ΔT_{JD} must fall between the minimum and maximum Julian date. The block **Epoch** parameter controls the number of block inputs. See the **Ephemerides** parameter for the minimum and maximum Julian dates.
Second (Optional)	Scalar	ΔT_{JD} — Elapsed Julian time (ΔT_{JD}) between the fixed date and the ephemeris time. The sum of $T0_{JD}$ and ΔT_{JD} must fall between the minimum and maximum Julian date. See the **Ephemerides** parameter for the minimum and maximum Julian dates.

Output	Dimension Type	Description
First	Vector	Barycenter position (X_{ICRF1}) of the **Target** object relative to the barycenter of the **Center** object. Units are km or astronomical units (AU). **Tip** This block outputs the barycenter position in Earth-centered inertial (ECI) coordinates. To convert these coordinates to Earth-centered Earth-fixed (ECEF), use the Direction Cosine Matrix ECI to ECEF block.
Second (Optional)	Vector	Velocity (V_{ICRF}) of the barycenter of the **Target** object relative to the barycenter of the **Center** 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

Direction Cosine Matrix ECI to ECEF | Earth Nutation | Julian Date Conversion | Moon Libration

External Websites

`http://ssd.jpl.nasa.gov/?planet_eph_export`

Introduced in R2013a

Precision Pilot Model

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] on page 4-458). 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.
τ	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.
ω_N	Undamped frequency for the neuromuscular system.
ζ_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(gcb,'pade','3')
```

Parameters

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, $T_I >> T_L$

Option (Controlled Element Transfer Function)	Transfer Function of Controlled Element (Y_c)	Transfer Function of Pilot (Y_p)
Rate or velocity	$\frac{K_c}{s}$	1
Acceleration	$\frac{K_c}{s^2}$	Lead-lag, $T_L >> T_I$
Second order	$\frac{K_c \omega_n{}^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$	Lead-lag if $\omega_m << 2/\tau$. Lag-lead if $\omega_m >> 2/\tau$.

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.

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.

Controlled element undamped natural frequency (rad/s)

Specifies the controlled element undamped natural frequency in rad/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 188, 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.

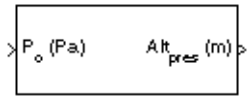
See Also

Crossover Pilot Model | Tustin Pilot Model

Introduced in R2012b

Pressure Altitude

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.

Parameters

Units

Specifies the input units:

Units	Pstatic	Alt_p
`Metric (MKS)`	Pascal	Meters
`English`	Pound force per square inch	Feet

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.

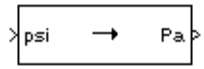
See Also

COESA Atmosphere Model

Introduced before R2006a

Pressure Conversion

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 unit** and the **Final unit** lists.

Parameters

Initial unit

Specifies the input units.

Final unit

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

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

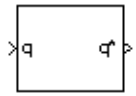
Temperature Conversion

Velocity Conversion

Introduced before R2006a

Quaternion Conjugate

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

$$q' = q_0 - \mathbf{i}q_1 - \mathbf{j}q_2 - \mathbf{k}q_3.$$

Inputs and Outputs

Input	Dimension Type	Description
First	Quaternion or vector	Contains quaternions in the form of [q_0, r_0, ... , q_1, r_1, ... , q_2, r_2, ... , q_3, r_3, ...].

Output	Dimension Type	Description
First	Quaternion conjugate or vector	Contains quaternion conjugates in the form of [q_0', r_0', ... , q_1', r_1', ... , q_2', r_2', ... , q_3', r_3', ...].

References

Stevens, Brian L., Frank L. Lewis, *Aircraft Control and Simulation*, Wiley–Interscience, 2nd Edition.

See Also

Quaternion Division

Quaternion Inverse

Quaternion Modulus

Quaternion Multiplication

Quaternion Norm

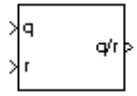
Quaternion Normalize

Quaternion Rotation

Introduced before R2006a

Quaternion Division

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

$$t_0 = \frac{(r_0 q_0 + r_1 q_1 + r_2 q_2 + r_3 q_3)}{r_0^2 + r_1^2 + r_2^2 + r_3^2}$$
$$t_1 = \frac{(r_0 q_1 - r_1 q_0 - r_2 q_3 + r_3 q_2)}{r_0^2 + r_1^2 + r_2^2 + r_3^2}$$
$$t_2 = \frac{(r_0 q_2 + r_1 q_3 - r_2 q_0 - r_3 q_1)}{r_0^2 + r_1^2 + r_2^2 + r_3^2}$$
$$t_3 = \frac{(r_0 q_3 - r_1 q_2 + r_2 q_1 - r_3 q_0)}{r_0^2 + r_1^2 + r_2^2 + r_3^2}$$

Inputs and Outputs

Input	Dimension Type	Description
First	Quaternion or vector	Contains quaternions in the form of [q_0, p_0, ..., q_1, p_1, ... , q_2, p_2, ... , q_3, p_3, ...].
Second	Quaternion or vector	Contains quaternions in the form of [s_0, r_0, ..., s_1, r_1, ... , s_2, r_2, ... , s_3, r_3, ...].

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.

References

Stevens, Brian L., Frank L. Lewis, *Aircraft Control and Simulation*, Wiley–Interscience, 2nd Edition.

See Also

Quaternion Conjugate

Quaternion Inverse

Quaternion Modulus

Quaternion Multiplication

Quaternion Norm

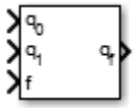
Quaternion Normalize

Quaternion Rotation

Introduced before R2006a

Quaternion Interpolation

Quaternion interpolation between two quaternions



Library

Utilities/Math Operations

Description

The Quaternion Interpolation block calculates the quaternion interpolation between two normalized quaternions by an interval fraction.

The two normalized quaternions are the two extremes between which the block calculates the quaternion.

Parameters

Methods

Specify the quaternion interpolation method to calculate the quaternion interpolation. These methods have different rotational velocities, depending on the interval fraction. For more information on interval fractions, see `http://web.mit.edu/2.998/www/QuaternionReport1.pdf`.

- SLERP

  Quaternion slerp. Spherical linear quaternion interpolation method.

  $$Slerp(p,q,h) = p(p^{*}q)^{h} \text{ with } h \in [0,1].$$

- LERP

Quaternion lerp. Linear quaternion interpolation method.

$$LERP(p,q,h) = p(1-h) + qh \text{ with } h \in [0,1].$$

- NLERP

  Normalized quaternion linear interpolation method.

  With $r = LERP(p,q,h)$, $NLERP(p,q,h) = \frac{r}{|r|}$.

Action for out-of-range input

Specifies whether out-of-range input causes a warning, error, or no action.

Inputs and Outputs

Input	Dimension Type	Description
First	4-by-1 or 1-by-4 vector	First normalized quaternion for which to calculate the interpolation. This quaternion must be a normalized quaternion
Second	4-by-1 or 1-by-4 vector	Second normalized quaternion for which to calculate the interpolation. This quaternion must be a normalized quaternion.
Third	Scalar	Interval fraction by which to calculate the quaternion interpolation . This value varies between 0 and 1. It represents the intermediate rotation of the quaternion to be calculated. This fraction affects the interpolation method rotational velocities.

Output	Dimension Type	Description
First	double	Natural logarithm of quaternion.

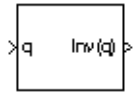
References

http://web.mit.edu/2.998/www/QuaternionReport1.pdf

Introduced in R2016a

Quaternion Inverse

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

$$q^{-1} = \frac{q_0 - \mathbf{i}q_1 - \mathbf{j}q_2 - \mathbf{k}q_3}{q_0^2 + q_1^2 + q_2^2 + q_3^2}.$$

Inputs and Outputs

Input	Dimension Type	Description
First	Quaternion or vector	Contains quaternions in the form of [q_0, r_0, ..., q_1, r_1, ... , q_2, r_2, ... , q_3, r_3, ...].

Output	Dimension Type	Description
First	Quaternion inverse or vector	Contains quaternion inverse or vector of quaternion inverses.

References

Stevens, Brian L., Frank L. Lewis, *Aircraft Control and Simulation*, Wiley–Interscience, 2nd Edition.

See Also

Quaternion Conjugate

Quaternion Division

Quaternion Modulus

Quaternion Multiplication

Quaternion Norm

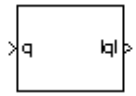
Quaternion Normalize

Quaternion Rotation

Introduced before R2006a

Quaternion Modulus

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

$$|q| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}$$

Inputs and Outputs

Input	Dimension Type	Description
First	Quaternion or vector	Contains quaternions in the form of [q_0, r_0, ..., q_1, r_1, ... , q_2, r_2, ... , q_3, r_3, ...].

Output	Dimension Type	Description
First	Quaternion modulus or vector	Contains quaternion modulus or vector of quaternion modulus in the form of [$\|q\|$, $\|r\|$, ...].

References

Stevens, Brian L., Frank L. Lewis, *Aircraft Control and Simulation*, Wiley–Interscience, 2nd Edition.

See Also

Quaternion Conjugate

Quaternion Division

Quaternion Inverse

Quaternion Multiplication

Quaternion Norm

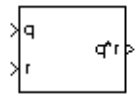
Quaternion Normalize

Quaternion Rotation

Introduced before R2006a

Quaternion Multiplication

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

$$\begin{aligned} t_0 &= (r_0 q_0 - r_1 q_1 - r_2 q_2 - r_3 q_3) \\ t_1 &= (r_0 q_1 + r_1 q_0 - r_2 q_3 + r_3 q_2) \\ t_2 &= (r_0 q_2 + r_1 q_3 + r_2 q_0 - r_3 q_1) \\ t_3 &= (r_0 q_3 - r_1 q_2 + r_2 q_1 + r_3 q_0) \end{aligned}$$

Inputs and Outputs

Input	Dimension Type	Description
First	Quaternion or vector	Contains quaternions in the form of [q_0, p_0, ..., q_1, p_1, ... , q_2, p_2, ... , q_3, p_3, ...].
Second	Quaternion or vector	Contains quaternions in the form of [s_0, r_0, ..., s_1, r_1, ... , s_2, r_2, ... , s_3, r_3, ...].

Output	Dimension Type	Description
First	Quaternion product or vector	Contains quaternion product or vector of quaternion products.

References

Stevens, Brian L., Frank L. Lewis, *Aircraft Control and Simulation*, Wiley–Interscience, 2nd Edition.

See Also

Quaternion Conjugate

Quaternion Division

Quaternion Inverse

Quaternion Modulus

Quaternion Norm

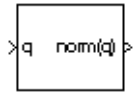
Quaternion Normalize

Quaternion Rotation

Introduced before R2006a

Quaternion Norm

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

$$norm(q) = q_0^2 + q_1^2 + q_2^2 + q_3^2$$

Inputs and Outputs

Input	Dimension Type	Description
First	Quaternion or vector	Contains quaternions in the form of [q_0, r_0, ..., q_1, r_1, ... , q_2, r_2, ... , q_3, r_3, ...].

Output	Dimension Type	Description
First	Quaternion norm or vector	Contains quaternion norm or vector of quaternion norms in the form of [*norm*(q), *norm*(r), ...].

References

Stevens, Brian L., Frank L. Lewis, *Aircraft Control and Simulation*, Wiley–Interscience, 2nd Edition.

See Also

Quaternion Conjugate

Quaternion Division

Quaternion Inverse

Quaternion Modulus

Quaternion Multiplication

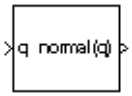
Quaternion Normalize

Quaternion Rotation

Introduced before R2006a

Quaternion Normalize

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

$$normal(q) = \frac{q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3}{\sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}}.$$

Inputs and Outputs

Input	Dimension Type	Description
First	Quaternion or vector	Contains quaternions in the form of [q_0, r_0, ..., q_1, r_1, ... , q_2, r_2, ... , q_3, r_3, ...].

Output	Dimension Type	Description
First	Normalized quaternion or vector	Contains normalized quaternion or vector of normalized quaternions.

References

Stevens, Brian L., Frank L. Lewis, *Aircraft Control and Simulation*, Wiley–Interscience, 2nd Edition.

See Also

Quaternion Conjugate

Quaternion Division

Quaternion Inverse

Quaternion Modulus

Quaternion Multiplication

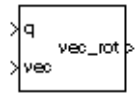
Quaternion Norm

Quaternion Rotation

Introduced before R2006a

Quaternion Rotation

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

$$v' = \begin{bmatrix} v_1' \\ v_2' \\ v_3' \end{bmatrix} = \begin{bmatrix} (1-2q_2^2-2q_3^2) & 2(q_1q_2+q_0q_3) & 2(q_1q_3-q_0q_2) \\ 2(q_1q_2-q_0q_3) & (1-2q_1^2-2q_3^2) & 2(q_2q_3+q_0q_1) \\ 2(q_1q_3+q_0q_2) & 2(q_2q_3-q_0q_1) & (1-2q_1^2-2q_2^2) \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

For more information, see Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors.

Inputs and Outputs

Input	Dimension Type	Description
First	Quaternion or vector	Contains quaternions in the form of [q_0, r_0, ..., q_1, r_1, ... , q_2, r_2, ... , q_3, r_3, ...].
Second	Vector	Contains vector or vector of vectors in the form of [v_1, u_1, ... , v_2, u_2, ... , v_3, u_3, ...].

Output	Dimension Type	Description
First	Rotated quaternion or vector	Contains rotated vector or vector of rotated vectors.

References

Stevens, Brian L., Frank L. Lewis, *Aircraft Control and Simulation*, Wiley–Interscience, 2nd Edition.

Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors

See Also

Quaternion Conjugate

Quaternion Division

Quaternion Inverse

Quaternion Modulus

Quaternion Multiplication

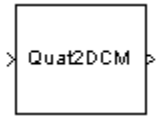
Quaternion Norm

Quaternion Normalize

Introduced before R2006a

Quaternions to Direction Cosine Matrix

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:

$$P' = qPq^c$$
$$q = q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3$$
$$q^c = q_0 - \mathbf{i}q_1 - \mathbf{j}q_2 - \mathbf{k}q_3$$
$$P = 0 + \mathbf{i}x + \mathbf{j}y + \mathbf{k}z$$

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.

$$DCM = \begin{bmatrix} (q_0^2 + q_1^2 - q_2^2 - q_3^2) & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & (q_0^2 - q_1^2 + q_2^2 - q_3^2) & 2(q_2q_3 + q_0q_1) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & (q_0^2 - q_1^2 - q_2^2 + q_3^2) \end{bmatrix}$$

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

Introduced before R2006a

Quaternions to Rodrigues

Convert quaternion to Euler-Rodrigues vector
Library: Utilities / Axes Transformations

Description

The Quaternions to Rodrigues block converts the 4-by-1 quaternion to the 3-element Euler-Rodrigues vector.

Ports

Input

`Quaternion` — Quaternion
4-by-1 matrix

Quaternion from which to determine Euler-Rodrigues vector. Quaternion scalar is the first element.

Data Types: `double`

Output

`rod` — Euler-Rodrigues vector
3-element vector

Euler-Rodrigues vector determined from the quaternion.

Data Types: `double`

Algorithms

- An Euler-Rodrigues vector $\vec{b}$ represents a rotation by integrating a direction cosine of a rotation axis with the tangent of half the rotation angle as follows:

  $$\vec{b} = \begin{bmatrix} b_x & b_y & b_z \end{bmatrix}$$

  where:

  $$b_x = \tan\left(\frac{1}{2}\theta\right)s_x,$$
  $$b_y = \tan\left(\frac{1}{2}\theta\right)s_y,$$
  $$b_z = \tan\left(\frac{1}{2}\theta\right)s_z$$

  are the Rodrigues parameters. Vector $\vec{s}$ represents a unit vector around which the rotation is performed. Due to the tangent, the rotation vector is indeterminate when the rotation angle equals ±pi radians or ±180 deg. Values can be negative or positive.

References

[1] Dai, J.S. "Euler-Rodrigues formula variations, quaternion conjugation and intrinsic connections." *Mechanism and Machine Theory*, 92, 144-152. Elsevier, 2015.

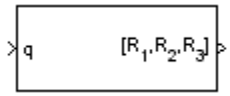
See Also

Direction Cosine Matrix to Rodrigues | Rodrigues to Direction Cosine Matrix | Rodrigues to Quaternions | Rodrigues to Rotation Angles | Rotation Angles to Rodrigues

Introduced in R2017a

Quaternions to Rotation Angles

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:

$$\begin{aligned}\phi &= \text{atan}(DCM(2,3), DCM(3,3)) \\ &= \text{atan}(2(q_2q_3 + q_0q_1), (q_0^2 - q_1^2 - q_2^2 + q_3^2)) \\ \theta &= \text{asin}(-DCM(1,3)) \\ &= \text{asin}(-2(q_1q_3 - q_0q_2)) \\ \psi &= \text{atan}(DCM(1,2), DCM(1,1)) \\ &= \text{atan}(2(q_1q_2 + q_0q_3), (q_0^2 + q_1^2 - q_2^2 - q_3^2))\end{aligned}$$

where Ψ is R1, Θ is R2, and Φ is R3.

Parameters

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-1 vector	Contains the rotation angles, in radians.

Assumptions and Limitations

For the 'ZYX', 'ZXY', 'YXZ', 'YZX', 'XYZ', and 'XZY' rotations, the block generates an R2 angle that lies between ±pi/2 radians, and R1 and R3 angles that lie between ±pi radians.

For the 'ZYZ', 'ZXZ', 'YXY', 'YZY', 'XYX', and 'XZX' rotations, the block generates an R2 angle that lies between 0 and pi radians, and R1 and R3 angles that lie between ±pi radians. However, in the latter case, when R2 is 0, R3 is set to 0 radians.

See Also

Direction Cosine Matrix to Rotation Angles

Direction Cosine Matrix to Quaternions

Quaternions to Direction Cosine Matrix

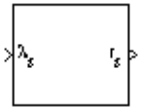
Rotation Angles to Direction Cosine Matrix

Rotation Angles to Quaternions

Introduced in R2007b

Radius at Geocentric Latitude

Estimate radius of ellipsoid planet at geocentric latitude

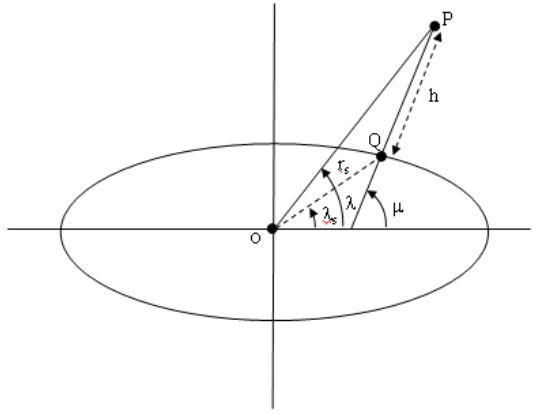


Library

Flight Parameters

Description

The Radius at Geocentric Latitude block estimates the radius (r_s) of an ellipsoid planet at a particular geocentric latitude (λ_s).



The following equation estimates the ellipsoid radius (r_s) using flattening $(\bar{f})$, geocentric latitude $(\bar{\lambda}_s)$, and equatorial radius $(\bar{R})$.

	MIL-F-8785C
Longitudinal	
Lateral	
$H_r(s)$	$\frac{\mp\frac{s}{V}}{\left(1+\left(\frac{3b}{\pi V}\right)s\right)}\cdot H_v(s)$
Vertical	
$H_w(s)$	$\frac{\sigma_w\sqrt{\frac{1}{\pi}\cdot\frac{L_w}{V}}\left(1+2.7478\frac{L_w}{V}s+0.3398\left(\frac{L_w}{V}\right)^2 s^2\right)}{1+2.9958\frac{L_w}{V}s+1.9754\left(\frac{L_w}{V}\right)^2 s^2+0.1539\left(\frac{L_w}{V}\right)^3 s^3}$

	MIL-F-8785C
$H_q(s)$	$\frac{\pm\frac{s}{V}}{\left(1+\left(\frac{4b}{\pi V}\right)s\right)}\cdot H_w(s)$

	MIL-HDBK-1797
Longitudinal	
$H_u(s)$	$\frac{\sigma_u\sqrt{\frac{2}{\pi}\cdot\frac{L_u}{V}}\left(1+0.25\frac{L_u}{V}s\right)}{1+1.357\frac{L_u}{V}s+0.1987\left(\frac{L_u}{V}\right)^2 s^2}$
$H_p(s)$	$\sigma_w\sqrt{\frac{0.8}{V}}\cdot\frac{\left(\frac{\pi}{4b}\right)^{1/6}}{(2L_w)^{1/3}\left(1+\left(\frac{4b}{\pi V}\right)s\right)}$
Lateral	
$H_v(s)$	$\frac{\sigma_v\sqrt{\frac{1}{\pi}\cdot\frac{2L_v}{V}}\left(1+2.7478\frac{2L_v}{V}s+0.3398\left(\frac{2L_v}{V}\right)^2 s^2\right)}{1+2.9958\frac{2L_v}{V}s+1.9754\left(\frac{2L_v}{V}\right)^2 s^2+0.1539\left(\frac{2L_v}{V}\right)^3 s^3}$
$H_r(s)$	$\frac{\mp\frac{s}{V}}{\left(1+\left(\frac{3b}{\pi V}\right)s\right)}\cdot H_v(s)$
Vertical	

	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
$L_w = h$ $L_u = L_v = \frac{h}{(0.177+0.000823h)^{1.2}}$	$2L_w = h$ $L_u = 2L_v = \frac{h}{(0.177+0.000823h)^{1.2}}$

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.

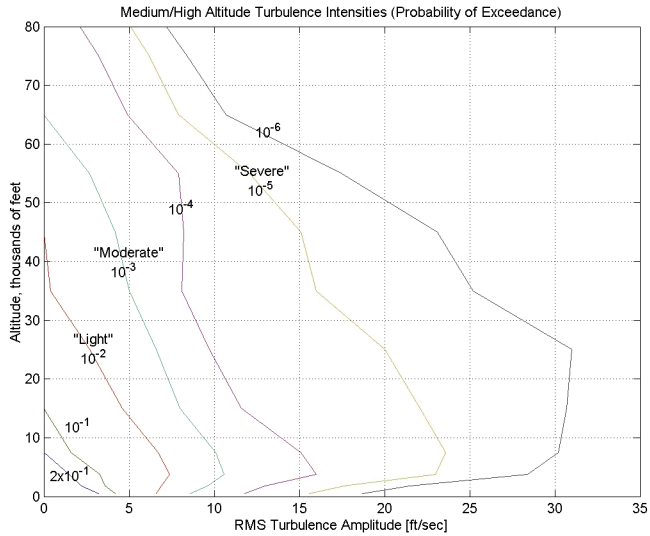
Medium/High Altitudes (Altitude > 2000 feet)

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
$L_u = L_v = L_w = 2500$ ft	$L_u = 2L_v = 2L_w = 2500$ 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:



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.

Parameters

Units

Define the units of wind speed due to the turbulence.

Units	Wind Velocity	Altitude	Air Speed
`Metric (MKS)`	Meters/second	Meters	Meters/second

Units	Wind Velocity	Altitude	Air Speed
`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.
`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.

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.

Inputs and Outputs

Input	Dimension Type	Description
First	scalar	Contains the altitude in units selected.
Second	scalar	Contains the aircraft speed in units selected.
Third	3-by-3 matrix	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 factions (except altitude)

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, 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

Introduced in R2006b

WGS84 Gravity Model

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.

Parameters

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:

Units	Height	Gravity
`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**. The year must be after January 1, 2000 (2451545). 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 **Precessing 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 **Precessing 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 **Precessing reference frame** is selected. It is disabled if you select **Input Julian Date**.

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 or *M*-by-3 array	Contains the position in geodetic latitude, longitude and altitude, with units in degrees, degrees, and selected units of length respectively. Altitude must be less than 20,000.0 m (approximately 65,620.0 feet).
Second (Optional)	Scalar	Contains the Julian centuries that you specified.

Output	Dimension Type	Description
Output	Three-element vector or *M*-by-3 array	Gravity in the north-east-down (NED) coordinate system. The Taylor Series and Close Approximation methods output only normal gravity (down in the NED Coordinate system). The Exact method outputs 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.

The WGS84 Close Approximation model 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.

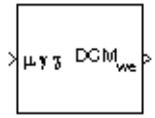
Reference

[1] NIMA TR8350.2: “Department of Defense World Geodetic System 1984, Its Definition and Relationship with Local Geodetic Systems.”

Introduced before R2006a

Wind Angles to Direction Cosine Matrix

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 (χ) to axes (ox_1, oy_1, oz_1)
2. A rotation about oy_1 through the flight path angle (γ) to axes (ox_2, oy_2, oz_2)
3. A rotation about ox_2 through the bank angle (μ) 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.

$$DCM_{we} = \begin{bmatrix} \cos\gamma\cos\chi & \cos\gamma\sin\chi & -\sin\gamma \\ (\sin\mu\sin\gamma\cos\chi - \cos\mu\sin\chi) & (\sin\mu\sin\gamma\sin\chi + \cos\mu\cos\chi) & \sin\mu\cos\gamma \\ (\cos\mu\sin\gamma\cos\chi + \sin\mu\sin\chi) & (\cos\mu\sin\gamma\sin\chi - \sin\mu\cos\chi & \cos\mu\cos\gamma \end{bmatrix}$$

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 ±90 degrees, and bank and heading angles that lie between ±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

Introduced before R2006a

Wind Angular Rates

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 (α), sideslip angle (β), 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.

$$\begin{bmatrix} p_w \\ q_w \\ r_w \end{bmatrix} = \begin{bmatrix} \cos\alpha\cos\beta & \sin\beta & \sin\alpha\cos\beta \\ -\cos\alpha\sin\beta & \cos\beta & -\sin\alpha\sin\beta \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} \begin{bmatrix} p_b - \dot{\beta}\sin\alpha \\ q_b - \dot{\alpha} \\ r_b + \dot{\beta}\cos\alpha \end{bmatrix}$$

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.

Input	Dimension Type	Description
		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)

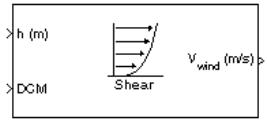
Simple Variable Mass 6DOF Wind (Quaternion)

Simple Variable Mass 6DOF Wind (Wind Angles)

Introduced before R2006a

Wind Shear Model

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.

$$u_w = W_{20} \frac{\ln\left(\frac{h}{z_0}\right)}{\ln\left(\frac{20}{z_0}\right)},\ 3ft < h < 1000ft$$

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.

Parameters

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 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)

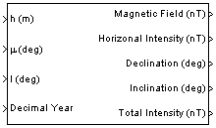
Dryden Wind Turbulence Model (Discrete)

Von Karman Wind Turbulence Model (Continuous)

Introduced before R2006a

World Magnetic Model 2000

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 2015, use International Geomagnetic Reference Field 12. For model years between 2015 and the start of 2020, use World Magnetic Model 2015.

Library

Environment/Gravity

Description

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).

Parameters

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.
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

Macmillian, 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`

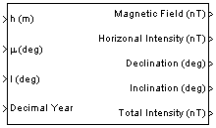
See Also

International Geomagnetic Reference Field 12, World Magnetic Model 2010

Introduced before R2006a

World Magnetic Model 2005

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 2015, use International Geomagnetic Reference Field 12. For model years between 2015 and the start of 2020, use World Magnetic Model 2015.

Library

Environment/Gravity

Description

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.

Parameters

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 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

Input	Dimension Type	Description
First		Contains the height, in selected units.
Second		Contains the latitude in degrees.
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. The reference frame is north-east-down (NED).
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.

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`

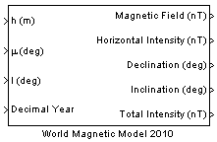
See Also

International Geomagnetic Reference Field 12, World Magnetic Model 2010

Introduced before R2006a

World Magnetic Model 2010

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.

Parameters

Units

Specifies the input and output units:

Units	Height	Magnetic Field	Horizontal Intensity	Total Intensity
`Metric (MKS)`	Meters	Nanotesla	Nanotesla	Nanotesla

Units	Height	Magnetic Field	Horizontal Intensity	Total Intensity
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

Input	Dimension Type	Description
First		Contains the height, in selected units.

Input	Dimension Type	Description
Second		Contains the latitude in degrees.
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`

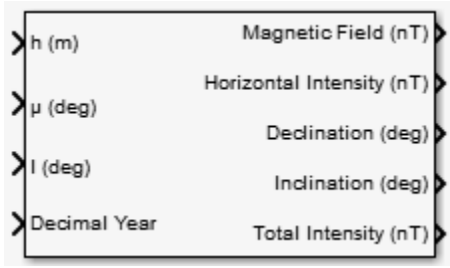
See Also

World Magnetic Model 2000, World Magnetic Model 2005

Introduced in R2010a

World Magnetic Model 2015

Calculate Earth's magnetic field at specific location and time using World Magnetic Model 2015 (WMM2015)



Library

Environment/Gravity

Description

The WMM2015 block implements the mathematical representation of the National Geospatial Intelligence Agency (NGA) World Magnetic Model 2015. The WMM2015 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.

Parameters

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 2015 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.
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, 2015:
		`%%%BEGIN CODE%%%`
		`dyear=decyear('21-March-2015','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 WMM2015 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 WMM2015 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

Introduced in R2015a

Zonal Harmonic Gravity Model

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 (Re) 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

Planet	Equatorial Radius (Re) in Meters	Gravitational Parameter (GM) in m^3/s^2	Zonal Harmonic Coefficients (J Values)
Moon	1738.0e3	4902.799e9	0.0002027
Neptune	24764e3	6.809e15	0.004
Saturn	60268.e3	3.794e16	[0.01645 0 -0.001]
Uranus	25559.e3	5.794e15	0.012
Venus	6052.0e3	3.257e14	0.000027

Parameters

Units

Specify the input units:

Units	Position	Equatorial Radius	Gravitational Parameter
`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.
- 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.

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 **Planet model** has a value of `Earth`, 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.

Introduced in R2009b

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^2 (m/s^2), kilometers/second^2 (km/s^2), (kilometers/hour)/second (km/h-s), g-unit (g)	inches/second^2 (in/s^2), feet/second^2 (ft/s^2), (miles/hour)/second (mph/s), g-unit (g)
Angle	radian (rad), degree (deg), revolution	radian (rad), degree (deg), revolution
Angular acceleration	radians/second^2 (rad/s^2), degrees/second^2 (deg/s^2), revolutions/minute (rpm), revolutions/second (rps)	radians/second^2 (rad/s^2), degrees/second^2 (deg/s^2), 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^3 (kg/m^3)	pound mass/foot^3 (lbm/ft^3), slug/foot^3 (slug/ft^3), pound mass/inch^3 (lbm/in^3)
Force	newton (N)	pound (lb)
Inertia	kilogram-meter^2 (kg-m^2)	slug-foot^2 (slug-ft^2)
Length	meter (m)	inch (in), foot (ft), mile (mi), nautical mile (nm)
Mass	kilogram (kg)	slug (slug), pound mass (lbm)
Pressure	Pascal (Pa)	pound/inch^2 (psi), pound/foot^2 (psf), atmosphere (atm)

Quantity	Metric (MKS)	English
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