Aerospace Blockset

For Use with Simulink®

- Modeling
- Simulation
- Implementation

User's Guide

Version 1

The MathWorks

How to Contact The MathWorks:

	www.mathworks.com	Web
	comp.soft-sys.matlab	Newsgroup
@	support@mathworks.com	Technical support
	suggest@mathworks.com	Product enhancement suggestions
	bugs@mathworks.com	Bug reports
	doc@mathworks.com	Documentation error reports
	service@mathworks.com	Order status, license renewals, passcodes
	info@mathworks.com	Sales, pricing, and general information
☎	508-647-7000	Phone
	508-647-7001	Fax
✉	The MathWorks, Inc.	Mail
	3 Apple Hill Drive	
	Natick, MA 01760-2098	

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

Notice

THE MATHWORKS PROGRAMS HAVE NOT BEEN TESTED OR CERTIFIED BY
ANY GOVERNMENT AGENCY OR INDUSTRY REGULATORY ORGANIZATION OR
ANY OTHER THIRD PARTY. THE PROGRAMS SHOULD NOT BE RELIED ON AS
THE SOLE BASIS TO SOLVE A PROBLEM WHOSE INCORRECT SOLUTION
COULD RESULT IN INJURY TO PERSON OR PROPERTY. THE PROGRAMS ARE
NOT DESIGNED NOR TESTED FOR A LEVEL OF RELIABILITY SUITABLE FOR
USE AS CRITICAL COMPONENTS IN ANY LIFE SUPPORT OR OTHER
INFORMATION SYSTEMS OR HARDWARE, THE FAILURE OF WHICH CAN
REASONABLY BE EXPECTED TO CAUSE DEATH OR PERSONAL INJURY OR
PROPERTY OR ENVIRONMENTAL DAMAGE. LICENSEE AGREES THAT PRIOR
TO USING, INCORPORATING OR DISTRIBUTING THE PROGRAMS IN ANY
PRODUCT, IT WILL THOROUGHLY TEST THE PRODUCT AND THE
FUNCTIONALITY OF THE PROGRAMS IN THAT PRODUCT AND BE SOLELY
RESPONSIBLE FOR ANY PROBLEMS OR FAILURES.

Contents

Case Studies

3

4

A

Getting Started

The Aerospace Blockset lets you model aerospace systems for use with Simulink® and MATLAB®.

What Is the Aerospace Blockset?

The Aerospace Blockset brings the full power of Simulink to aerospace system design, integration, and simulation by providing key aerospace subsystems and components in the adaptable Simulink block format. From environmental models to equations of motion, from gain scheduling to animation, the blockset gives you the core components to assemble a broad range of large aerospace system architectures rapidly and efficiently.

You can use the Aerospace Blockset and Simulink to develop your aerospace system concepts and to efficiently revise and test your models throughout the life cycle of your design. Use the Aerospace Blockset together with Real-Time Workshop® to automatically generate code for real-time execution in rapid prototyping and for hardware-in-the-loop systems.

What's in This Chapter

This chapter introduces you to the capabilities of the Aerospace Blockset and its relationship to other MathWorks products:

- "Required Products" on page 1-3
- "Opening the Aerospace Blockset in Simulink" on page 1-4

Required Products

The MathWorks provides several products that are especially relevant to the kinds of tasks you can perform with the Aerospace Blockset. In particular, the Aerospace Blockset requires these products:

- MATLAB 7
- Control System Toolbox 5.2
- Simulink 6

Virtual Reality-Based Visualization

The optional virtual reality-based visualization blocks in the Aerospace Blockset require the Virtual Reality Toolbox Version 4.0. The Virtual Reality Toolbox includes a default viewer, which works on all platforms.

You can also install the blaxxun Contact plug-in viewer, Version 4.4, for Web browsers. This plug-in is included with the Virtual Reality Toolbox and works on Windows platforms only. It requires Java-enabled Microsoft Internet Explorer 4.0, Netscape Navigator 4.0, or a later version Web browser.

For more information about any of these products

- Consult the online documentation for that product
- Visit the MathWorks Web site, at `www.mathworks.com`; see the "Products" section

Opening the Aerospace Blockset in Simulink

To get started with the Aerospace Blockset, you need to use Simulink. All the blocks in the Aerospace Blockset are designed for use together with the blocks in the Simulink libraries. This section describes how to open the Aerospace Blockset on Windows and on UNIX platforms:

"Opening the Aerospace Blockset on Windows Platforms" on page 1-4

"Opening the Aerospace Blockset on UNIX Platforms" on page 1-7

Opening the Aerospace Blockset on Windows Platforms

You can open the Aerospace Blockset from the Simulink Library Browser.

Opening the Simulink Library Browser

To start Simulink, click the icon in the MATLAB toolbar, or enter

```
simulink
```

at the command line.

The Simulink Libraries

The libraries in the Simulink Library Browser contain all the basic elements you need to construct a model. Look here for basic math operations, switches, connectors, simulation control elements, and other items that do not have a specific aerospace orientation.

Opening the Aerospace Blockset

On Windows platforms, the **Simulink Library Browser** opens when you start Simulink. The left pane contains a list of all the blocksets that you currently have installed.
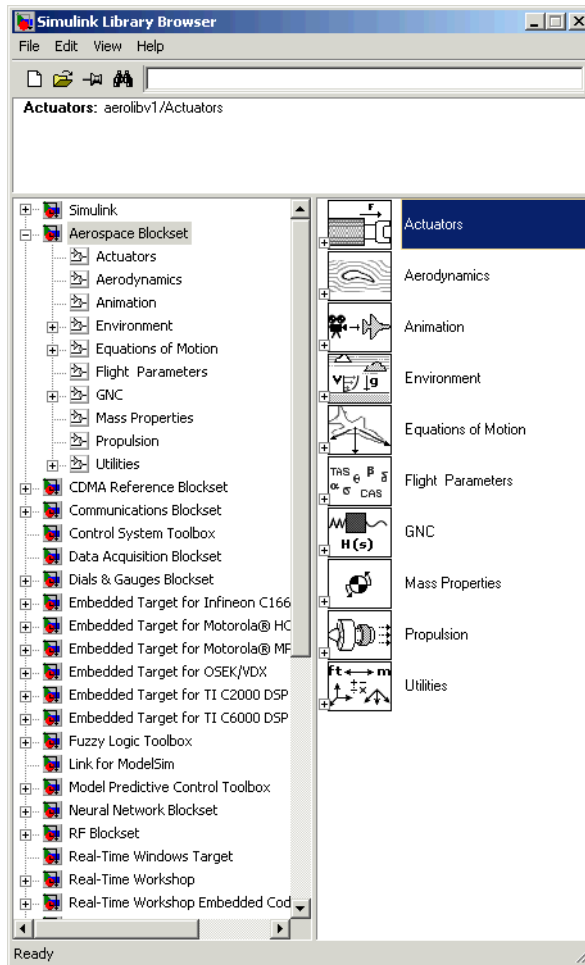
The first item in the list is the Simulink blockset itself, which is already expanded to show the available Simulink libraries. Click the ⊞ symbol to the left of any blockset name to expand the hierarchical list and display that blockset's libraries within the browser.

To open the Aerospace Blockset window from the MATLAB command line, enter

```
aerolib
```

Double-click any library in the window to display its contents. The following figure shows the **aerolib** library window.
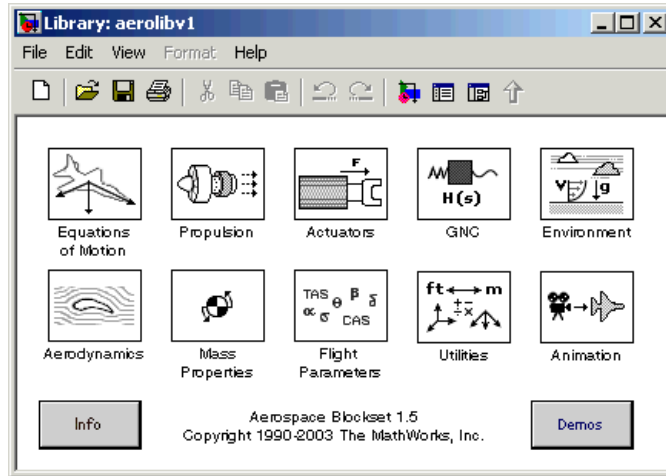
For a complete list of all the blocks in the Aerospace Blockset by library, see "Blocks — Categorical List" on page 4-2.

See the Simulink documentation for a complete description of the Simulink Library Browser.

Opening the Aerospace Blockset on UNIX Platforms

On UNIX platforms, the Simulink Library window opens when you start Simulink. To open the Aerospace Blockset, double-click the **Aerospace Blockset** icon to open the Aerospace Blockset.

To open the Aerospace Blockset window from the MATLAB command line, enter

```
aerolib
```

Double-click any library in the window to display its contents. The following figure shows the **aerolib** library window.
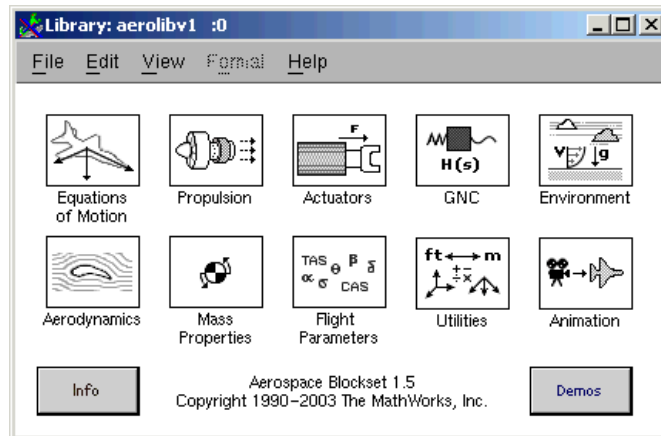
For a complete list of all the blocks in the Aerospace Blockset by library, see "Blocks — Categorical List" on page 4-2.

Running a Demo Model

This demo model uses some of the blocks in the Aerospace Blockset to simulate a three-degrees-of-freedom missile guidance system. You will see how the demo implements the Aerospace Blockset in conjunction with other Simulink blocks.

The demo model simulates a missile guidance system with a target acquisition and interception subsystem. The model implements a nonlinear representation of the rigid body dynamics of the missile airframe, including aerodynamic forces and moments. In addition, the missile autopilot was developed using the trimmed and linearized missile airframe, and the missile homing guidance systems regulates missile acceleration and measures the distance between the missile and the target.

Note For more information on the missile guidance model, see Chapter 3, "Case Studies."

What This Demo Illustrates

The missile guidance demo illustrates the following features of the Aerospace Blockset:

- Representing bodies and degrees of freedom with the Equations of Motion library blocks
- Using the Aerospace Blockset with other Simulink blocks
- Using the Aerospace Blockset with other Mathworks products like Stateflow
- Feeding in and feeding out Simulink signals to and from Aerospace Blockset blocks with Actuator and Sensor blocks
- Encapsulating groups of blocks into subsystems
- Visualizing and animating an aircraft with the Animation library blocks

Opening the Model

To open a Aerospace Blockset demo from the Help browser, open the Demos library in the Help browser by clicking the **Demos** tab in the **Help Navigator** pane on the left. Locate the demo in the list and open it. You can also open demos by entering the demo name at the command line.

Here is the general procedure for starting Aerospace Blockset demos from the **Start** button of the MATLAB desktop:

1 Click the **Start** button.

2 Select **Blocksets**, then **Aerospace**, and then **Demos**.

   This opens the MATLAB Help browser with **Demos** selected in the left **Help Navigator** pane.

3 Double-click **Three Degrees of Freedom Guided Missile** from the list of models in the list.

Alternatively, you can open the same MATLAB **Demos** window by entering demos at the MATLAB command line.

To get started quickly with this specific demo, you can enter aeroblk_guidance at the MATLAB command line.

The Block Diagram Model

The block diagram model opens in a model window.
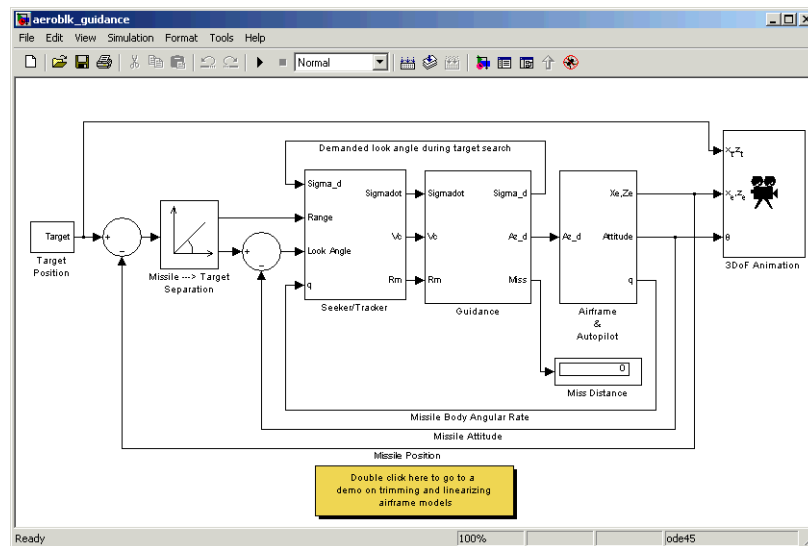
At the same time, a Stateflow® statechart appears that shows a chart for the guidance control processor.
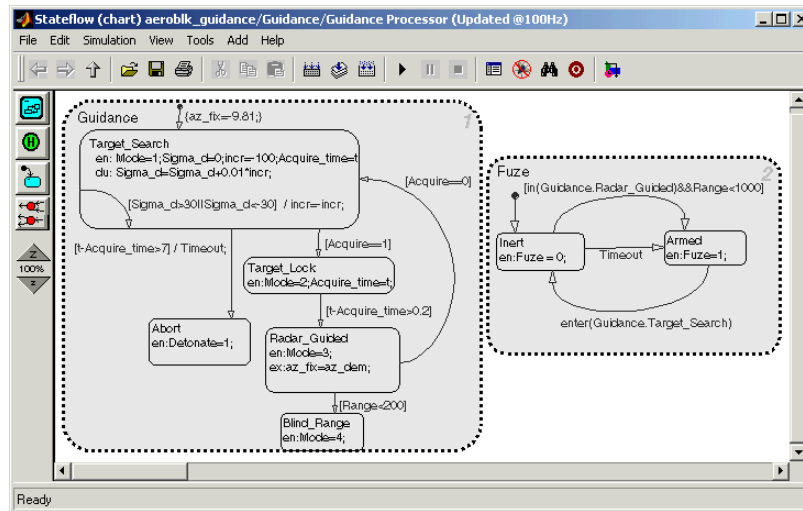
What the Model Contains

Note some features of the model:

• The Airframe & Autopilot subsystem implements the ISA Atmosphere Model block, the Incidence & Airspeed block, and the 3DoF (Body Axes) block, along with other Simulink blocks.

The airframe model is a nonlinear representation of rigid body dynamics. The aerodynamic forces and moments acting on the missile body are generated from coefficients that are nonlinear functions of both incidence and Mach number.
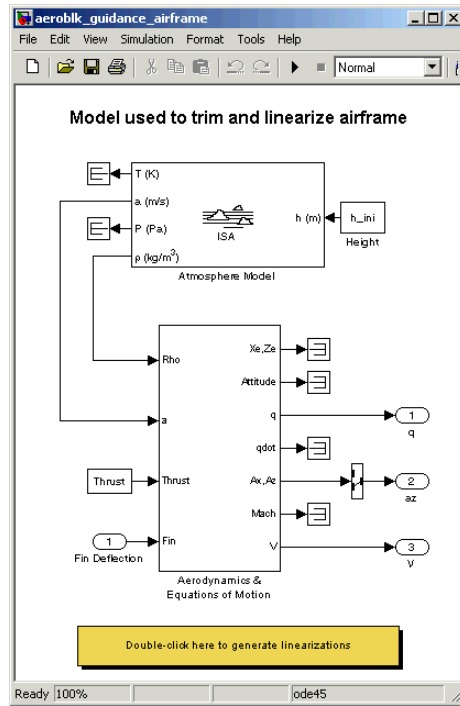
- The model implements the missile autopilot as a classical three-loop design using measurements from an accelerometer located ahead of the missile's center of gravity and from a rate gyro to provide additional damping.
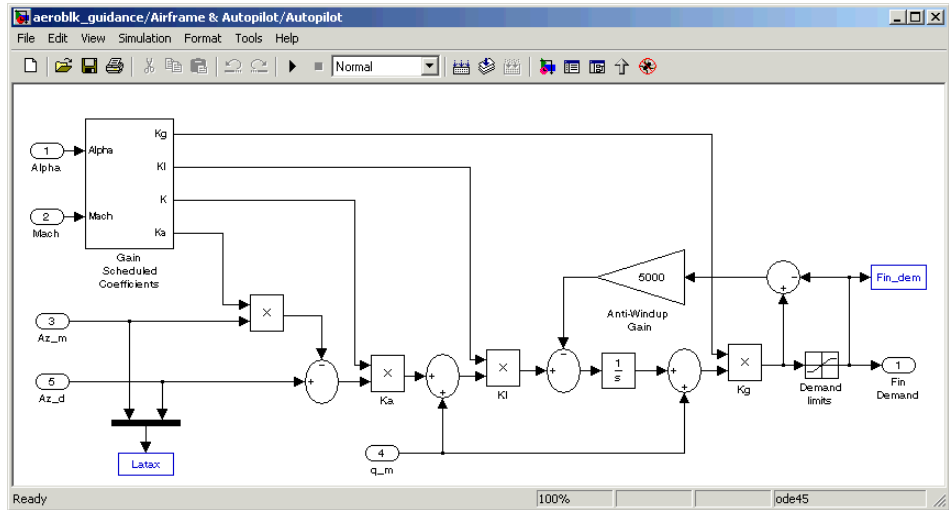
- The model implements the homing guidance system as two subsystems: the Guidance subsystem and the Seeker/Tracker subsystem.

  - The Guidance subsystem uses a Stateflow statechart to control the tracker directly by sending demands to the seeker gimbals.
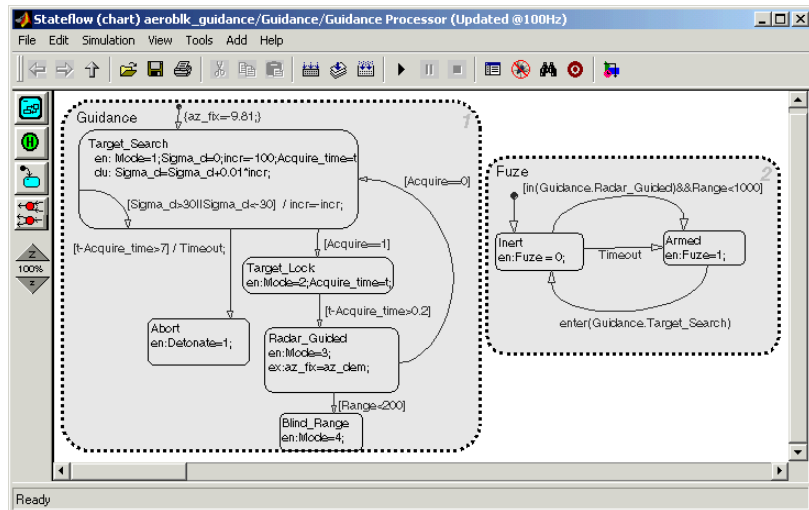
- The Seeker/Tracker subsystem consists of Simulink blocks that control the seeker gimbals to keep the seeker dish aligned with the target and provide the guidance law with an estimate of the sight line rate.
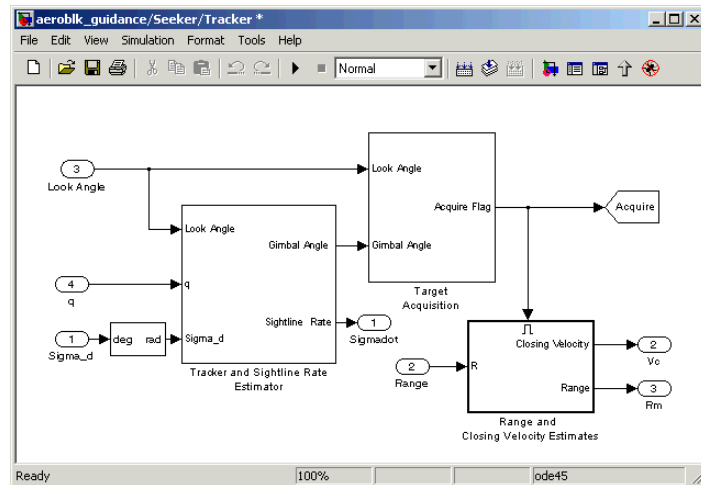
Running the Demo

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

1 Open the `aeroblk_guidance` demo.

2 From the **Simulation** menu, select **Start**. In Microsoft Windows, you can also click the **Start** button in the model window toolbar.

The simulation proceeds until the missile intercepts the target, which takes approximately 3 seconds. Once the interception has occurred, four scope figures open to display the following data:

a A three-dimensional animation of the missile and target interception course
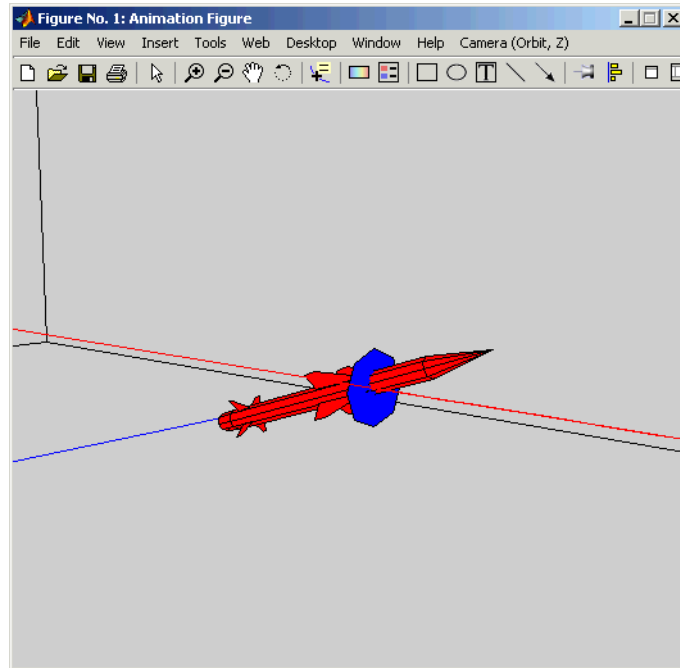
b Plots that measure flight parameters over time, including Mach number, fin demand, acceleration, and degree of incidence
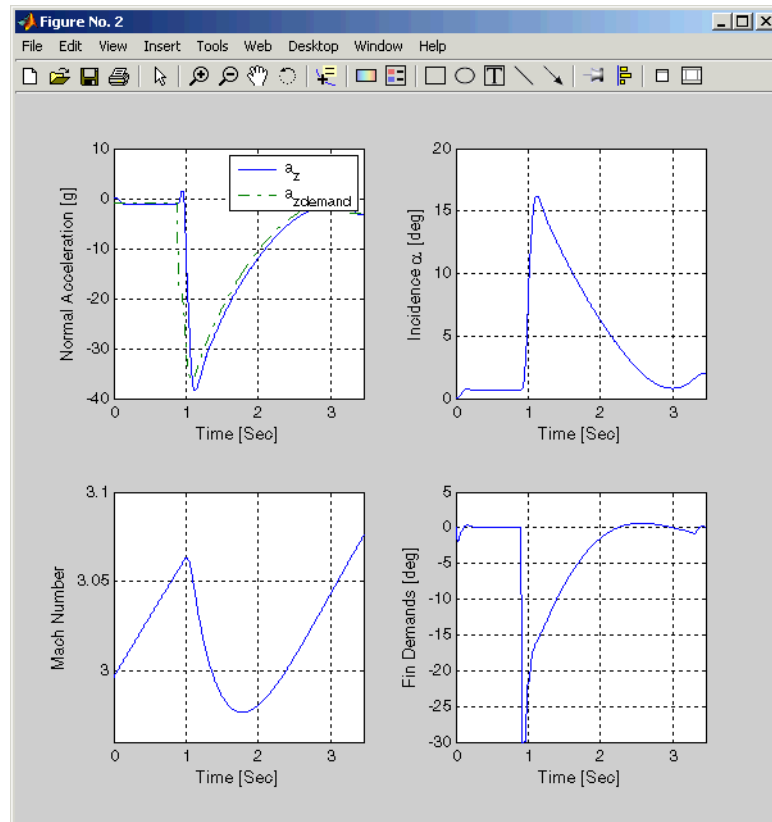
c A plot that measures gimbal versus true look angles
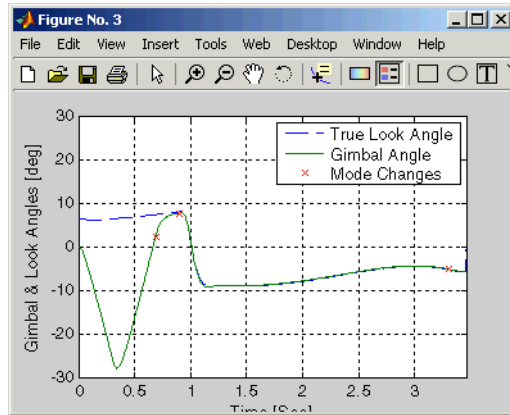
d A plot that measures missile and target trajectories
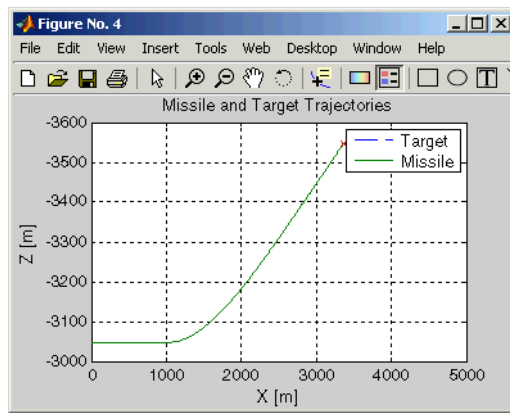
Modifying the Model

You can adjust model settings and examine the effects on simulation performance. Here are two modifications that you can try. The first modification adjusts the dynamic pressure for the simulation. The second modification changes the camera point of view for the interception animation.

Adjusting the Thrust

Like any Simulink model, you can adjust aerospace model parameters from the MATLAB workspace. To demonstrate this functionality, you will change the Thrust variable in the model and evaluate the results in the simulation. Follow these steps:

1 Open the `aeroblk_guidance` model in Simulink.

2 In the MATLAB desktop, find the Thrust variable in the **Workspace** panel.
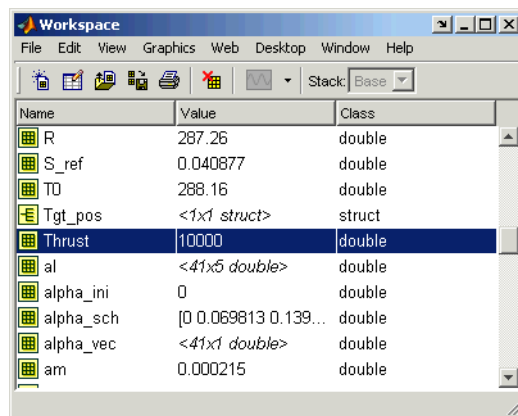
The Thrust variable is defined in the `aeroblk_guid_dat.m` file, which the `aeroblk_guidance` model uses to populate parameter and variable values. By default, the Thrust variable should be set to `10000`.

3 Single-click the Thrust variable to select it. To edit the value, right-click on the Thrust variable and select **Edit Value**. Change the value to `5000`.

Before you run the demo again, note the **Miss Distance** display in the **aeroblk_guidance** window.
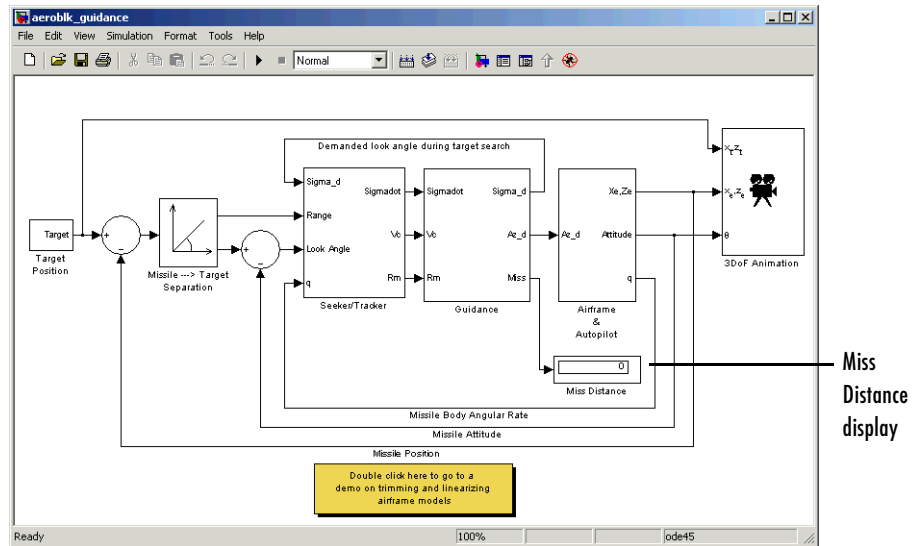
Start the demo, and after it finishes, note the miss distance display again. The miss distance should become greater than the original distance. You can experiment with different values in the `Thrust` variable and assess the effects on missile accuracy.

Changing the Animation Point of View

By default, the `aeroblk_guidance` model animation view is `Fly Alongside`, which means the view tracks with the missile's flight path. You can easily change the animation point of view by adjusting a parameter of the 3DoF Animation block:

1 Open the `aeroblk_guidance` model, and double-click the 3DoF Animation block. The **Block Parameters** dialog box appears.

Block Parameters: 3DoF Animation

3DoF_Animation (mask) (link)

Create a 3-D animated view of a three-degrees-of-freedom craft and its target, where X and Z target position (TargetPos), X and Z craft position (Xe,Ze), and craft attitude are inputs.

Display parameters are in the same units of length as the input parameters.

Parameters

Axes limits [xmin xmax ymin ymax zmin zmax]:

[0 5000 -2000 2000 -5050 -3050]

Time interval between updates:

0.03

Size of craft displayed:

1.8

Enter view: Fly alongside ⎯⎯⎯⎯⎯ Enter view

Position of camera [xc yc zc]:

[200 200 -50]

View angle:

20

☑ Enable animation

[OK] [Cancel] [Help] [Apply]

2 Change the view to Cockpit.

3 Click the **OK** button.

Run the demo again, and watch the animation. Instead of moving alongside the missile's flight path, the animation point of view lies in the cockpit. Upon target interception, the screen fills with blue, the target's color.
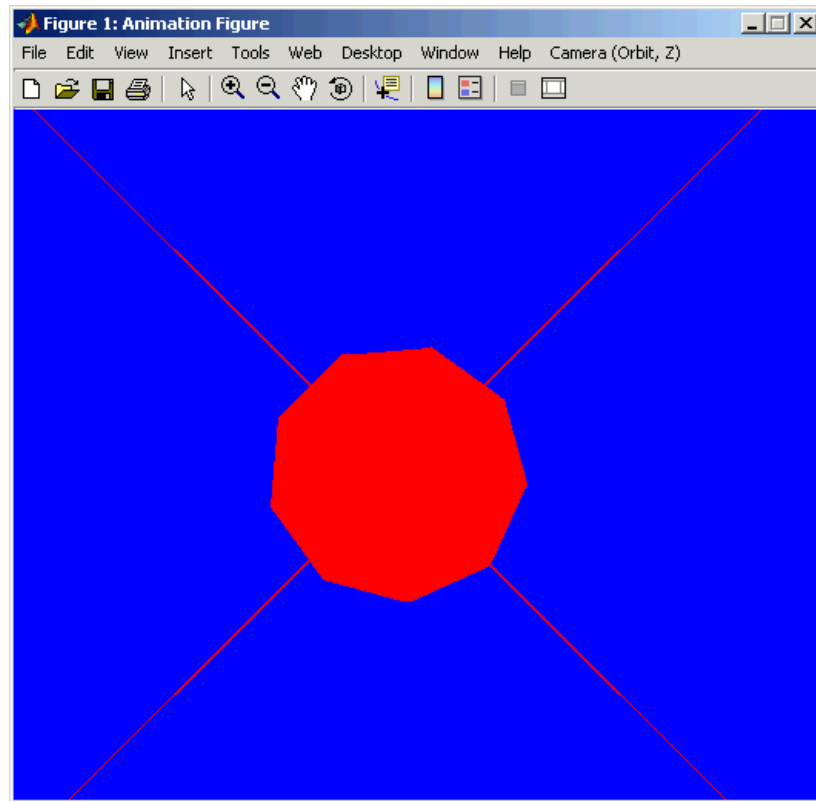
You can experiment with different views to watch the animation from different perspectives.

2

Using the Aerospace Blockset

Constructing a simple model with the Aerospace Blockset is easy to learn if you already know how to make Simulink models. If you are not already familiar with Simulink, please see the Simulink documentation.

Introducing the Aerospace Blockset Libraries (p. 2-2)

Overview of the Aerospace Blockset libraries

Creating Aerospace Models (p. 2-5)

Summary of the most important steps for building models with the Aerospace Blockset

Building a Simple Actuator System (p. 2-6)

A tutorial to model and simulate a simple actuator system

Introducing the Aerospace Blockset Libraries

The Aerospace Blockset is organized into hierarchical libraries of closely related blocks. The following sections summarize the blocks in each library. You can view the general Aerospace Blockset reference in Chapter 4, "Block Reference."

Note For more information on viewing the Aerospace Blockset, see Chapter 1, "Getting Started."

Actuators Library

The Actuators library provides blocks for representing linear and nonlinear actuators with saturation and rate limits.

Aerodynamics Library

The Aerodynamics library provides the Aerodynamic Forces and Moments block using the aerodynamic coefficients, dynamic pressure, center of gravity and center of pressure.

Animation Library

The Animation library provides the 3DoF Animation block and the 6DoF Animation block. Using the animation blocks, you can visualize flight paths and trajectories.

Environment Library

The Environment library provides blocks that simulate various aspects of an aircraft environment, such as atmosphere conditions, gravity, magnetic fields, and wind. The Environment library contains the Atmosphere, Gravity, and Wind sublibraries.

Atmosphere Sublibrary

The Atmosphere sublibrary provides general atmospheric models, such as ISA and COESA, and other blocks, including nonstandard day simulations, lapse rate atmosphere, and pressure altitude.

Gravity Sublibrary

The Gravity sublibrary provides blocks that calculate the gravity and magnetic fields for any point on the Earth.

Wind Sublibrary

The Wind sublibrary provides blocks for wind-related simulations, including turbulence, gust, shear, and horizontal wind.

Equations of Motion Library

The Equations of Motion library provides blocks for implementing the equations of motion to determine body position, velocity, attitude, and related values.The Equations of Motion library contains the 3DoF and 6DoF sublibraries.

3DoF Sublibrary

The 3DoF sublibrary provides blocks for implementing three-degrees-of-freedom equations of motion in your simulations, including custom variable mass models.

6DoF Sublibrary

The 6DoF sublibrary provides blocks for implementing six-degrees-of-freedom equations of motion in your simulations using Euler angles and quaternion representations.

Flight Parameters Library

The Flight Parameters library provides blocks for various parameters, including ideal airspeed correction, mach number, and dynamic pressure. The Flight Parameters library contains the Control and Guidance sublibraries.

Control Sublibrary

The Control sublibrary provides blocks for simulating various control types, such as one-dimensional, two-dimensional, and three-dimensional models.

Guidance Sublibrary

The Guidance sublibrary provides the Calculate Range block, which computes the range between two vehicles.

GNC Library

The GNC library provides blocks for creating control and guidance systems, including various controller models.

Mass Properties Library

The Mass Properties library provides blocks for simulating the center of gravity and inertia tensors.

Propulsion Library

The Propulsion library provides the Turbofan Engine System block, which simulates an engine system and controller.

Utilities Library

The Utilities library contains miscellaneous blocks useful in building models. The library contains the Axes Transformations, Math Operations, and Unit Conversions sublibraries.

Axes Transformations Sublibrary

The Axes Transformations sublibrary provides blocks for transforming axes of coordinate systems to different types, such as Euler angles to quaternions and vice versa.

Math Operations Sublibrary

The Math Operations sublibrary provides blocks for common mathematical and matrix operations, including sine and cosine generation and various 3-by-3 matrix operations.

Unit Conversions Sublibrary

The Unit Conversions sublibrary provides blocks for converting common measurement units from one system to another, such as converting acceleration from feet per second to meter per second and vice versa.

Creating Aerospace Models

Regardless of its complexity, you use the same procedure for creating an aerospace model as you would for creating any other Simulink model. Here are the basic steps:

1 *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 will select one block 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. Usually, a simulation block represents a part of the model and design functionality to be simulated, such as an actuator block, a mathematical operation, a block from the Aerospace Blockset, and so on.

   c Signal Routing blocks route signals from one point in a model to another. If you have two or more signals in your block, you will likely use a signal routing block in your models, including Mux blocks.

   d Sink blocks display or write model output. To see the results of the simulation, you must use a Sink block.

2 *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.

3 *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. For more information on connecting blocks, see the Simulink documentation.

4 *Encapsulate subsystems.* Systems made with the Aerospace Blockset can function as subsystems of larger, more complex models, like subsystems in normal Simulink models. For more information on subsystems, see the Simulink documentation.

Building a Simple Actuator System

In this tutorial, you drag, drop, and configure a few basic blocks to drive, simulate, and measure an actuator. The tutorial guides you through these aspects of model building:

- "Building the Model" on page 2-6
- "Running the Simulation" on page 2-15

At the end of the tutorial, you will have constructed a simple actuator model that measures the actuator's position in relation to a sine wave.

Building the Model

Simulink 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.
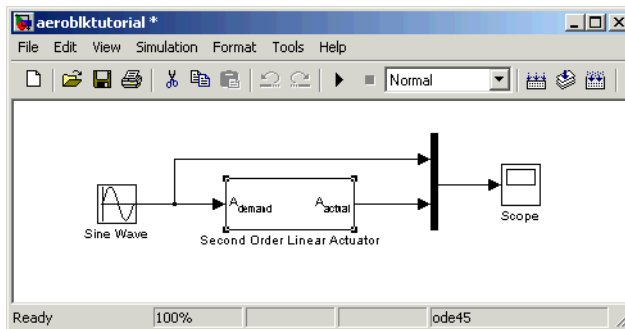
The following sections explain how to build a model on Windows and UNIX platforms:

- "Creating a Model on Windows Platforms" on page 2-7
- "Creating a Model on UNIX Platforms" on page 2-11

Creating a Model on Windows Platforms

1 Start Simulink.

Click the ![icon] button in the MATLAB toolbar or enter `simulink` at the MATLAB command line. The **Simulink Library Browser** appears.
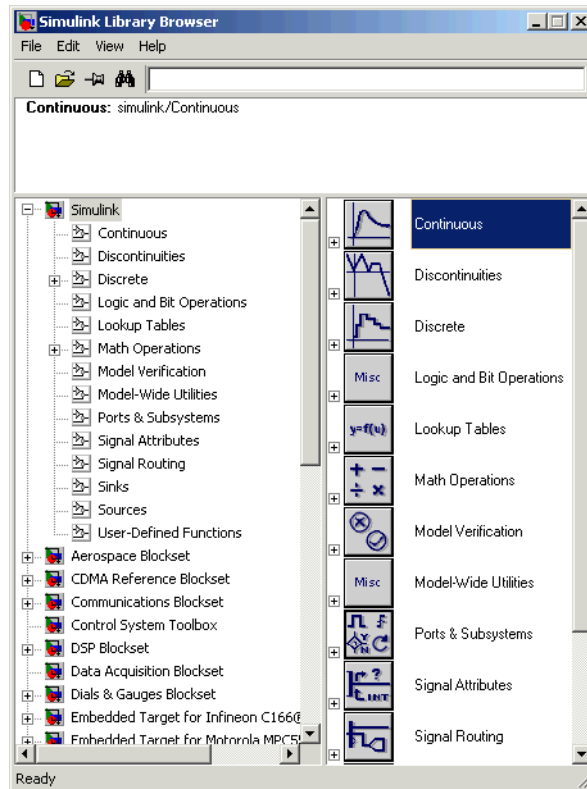
2 Open a new model.

Select **New -> Model** from the **File** menu in the Library Browser. A new model window appears on your screen.

3 Add a Sine Wave block to the model.

a Click **Sources** in the Library Browser to view the blocks in the Simulink Sources library.

b Drag the Sine Wave block from the Sources library into the new model window.

4 Add a Second Order Linear 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 Second Order Linear 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 Second Order Linear Actuator block. Release the mouse button.

  b Using the same technique, connect the output of the Second Order Linear 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/sec 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**.
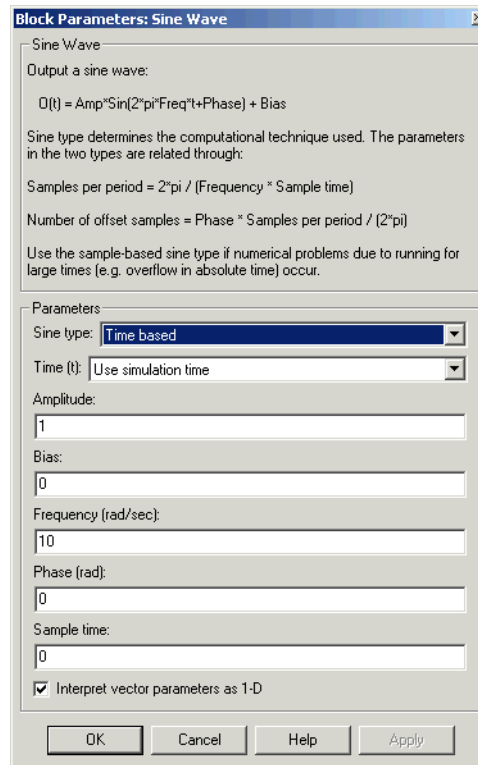
c Double-click the Second Order Linear Actuator block.

  In this example, the actuator has the default natural frequency of 150 rad/sec, 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**.
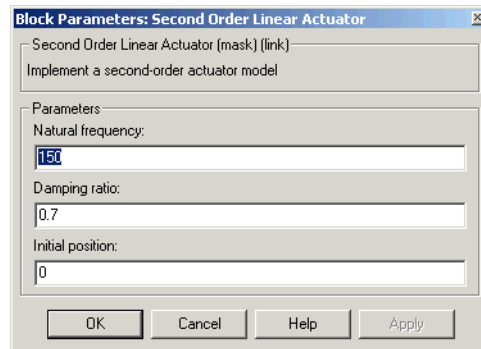
Creating a Model on UNIX Platforms

For this section, the screenshots were taken from an X Windows client in Microsoft Windows:

1 Start Simulink.

Enter `simulink` at the MATLAB command line. The Simulink Library window appears.

2 Open a new model.

Select **New -> Model** from the **File** menu in the Simulink Library window. A new model window appears on your screen.

3 Add a Sine Wave block to the model.

    a Double-click **Sources** in the Simulink Library window 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 Second Order Linear Actuator block to the model.

    a Double-click **Aerospace Blockset** in the Simulink Library browser. This opens the Aerospace Blockset block libraries.

    b In the Aerospace Blockset block libraries, click **Actuators** to view the blocks in the Actuator library.

    c Drag the Second Order Linear Actuator block into the model window.

5 Add a Mux block to the model.

    a Double-click **Signal Routing** in the Simulink Library to view the Signal Routing blocks.

    b Drag the Mux block from the Signal Routing library into the model window.

6 Add a Scope block to the model.

    a Double-click **Sinks** in the Simulink Library window 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 Second Order Linear Actuator block. Release the mouse button.

    b Using the same technique, connect the output of the Second Order Linear 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.

In this example, configure the block to generate a 10 rad/sec 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 Second Order Linear Actuator block.

For this example, the actuator has the default natural frequency of 150 rad/sec, 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**.

Block Parameters: Second Order Linear ... ✕

Second Order Linear Actuator (mask) (link)

Implement a second–order actuator model

Parameters

Natural frequency:

150

Damping ratio:

0.7

Initial position:

0

OK   Cancel   Help   Apply

Running the Simulation

You can now run the simulation block diagram 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 **Start** from the **Simulation** menu in the block diagram 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 it.

   b Change the frequency of the sinusoid. Try entering 1 or 20 in the **Frequency** field. Click **Apply** after entering each new value and 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. There are some parameters, however, like the Sine Wave **Frequency** parameter, that you can *tune* without terminating the simulation.

Running a Simulation from an M-File

You can also modify and run a Simulink simulation from within a MATLAB M-file. 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 "Running a Simulation Programmatically" in the Simulink documentation.

3

Case Studies

Missile Guidance System

This section explains how to design and simulate a three-degrees-of-freedom missile guidance system using Simulink and the Aerospace Blockset:

Missile Guidance System Model

To view the missile guidance system model, enter the following at the MATLAB command line.

```
aeroblk_guidance
```

The missile airframe and autopilot are contained in the Airframe & Autopilot subsystem. The Seeker/Tracker and Guidance subsystems model the homing guidance loop.
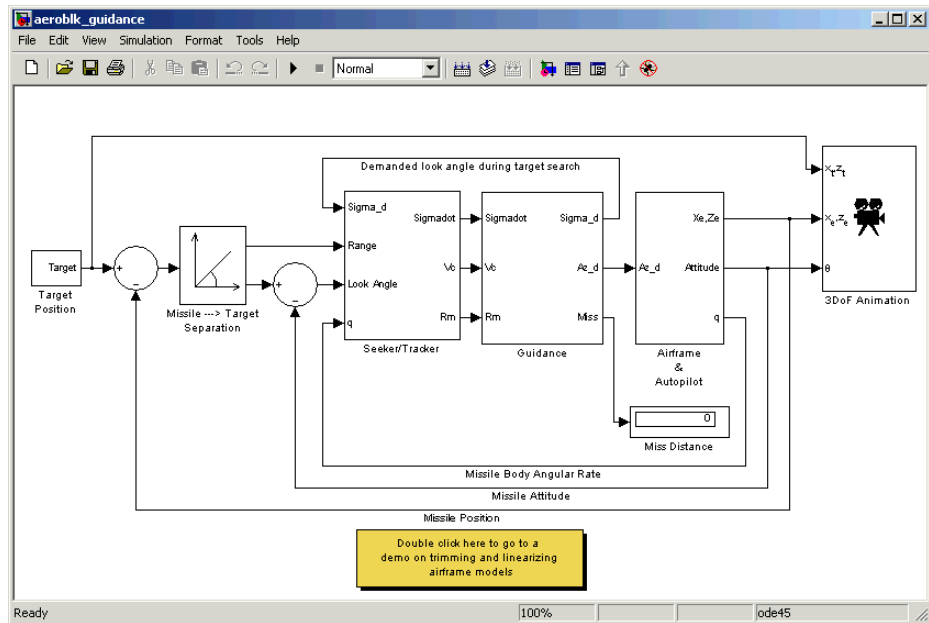
Modeling Airframe Dynamics

The model of the missile airframe in this demo uses advanced control methods applied to missile autopilot design [1], [2], [3]. The model represents a tail-controlled missile traveling between Mach 2 and Mach 4, at altitudes ranging between 3050 meters (10000 feet) and 18290 meters (60000 feet), and with typical angles of attack in the range of ±20 degrees.
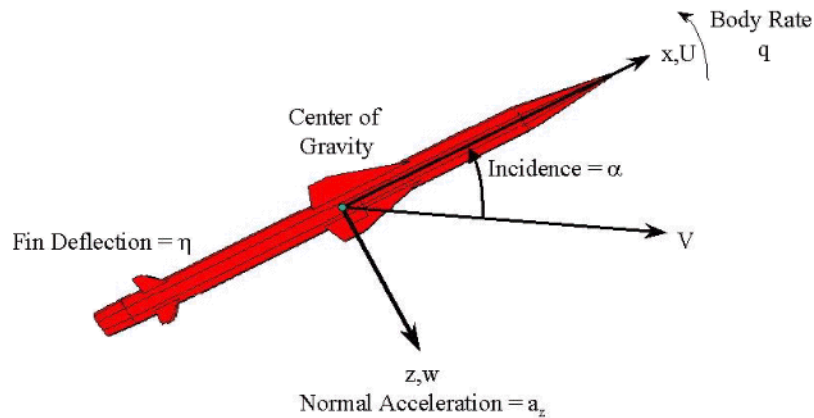
Missile Airframe Model

The core element of the model is a nonlinear representation of the rigid body dynamics of the airframe. The aerodynamic forces and moments acting on the missile body are generated from coefficients that are nonlinear functions of both incidence and Mach number. You can model these dynamics easily in the Simulink environment using the Aerospace Blockset.

The model of the missile airframe consists of two main components:

- "ISA Atmosphere Model Block" on page 3-5 calculates the change in atmospheric conditions with changing altitude.
- "Aerodynamics & Equations of Motion Subsystem" on page 3-8 calculates the magnitude of the forces and moments acting on the missile body and integrates the equations of motion.

To view the missile airframe model, enter the following in the MATLAB Command Window.

```
aeroblk_guidance_airframe
```
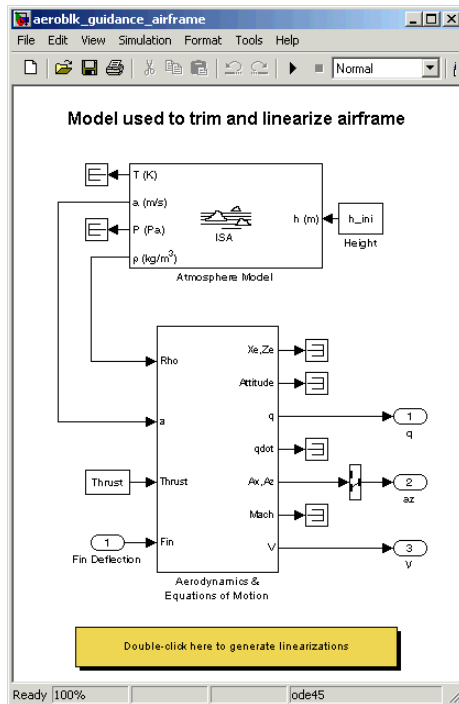

ISA Atmosphere Model Block

The ISA Atmosphere Model block is an approximation of the International Standard Atmosphere (ISA). This block consists of two sets of equations. One set of equations models is used for the troposphere region, and the other set of equations models is used for the lower stratosphere region. The troposphere region lies between sea level and 11000 meters (36089 feet). The ISA model assumes a linear temperature drop with increasing altitude in the troposphere region. The lower stratosphere region ranges between 11000 meters (36089 feet) and 20000 meters (65617 feet). The ISA models the stratosphere by assuming that the temperature remains constant in the lower stratosphere

region. The figure below displays how the speed of sound and the air density vary with altitude.
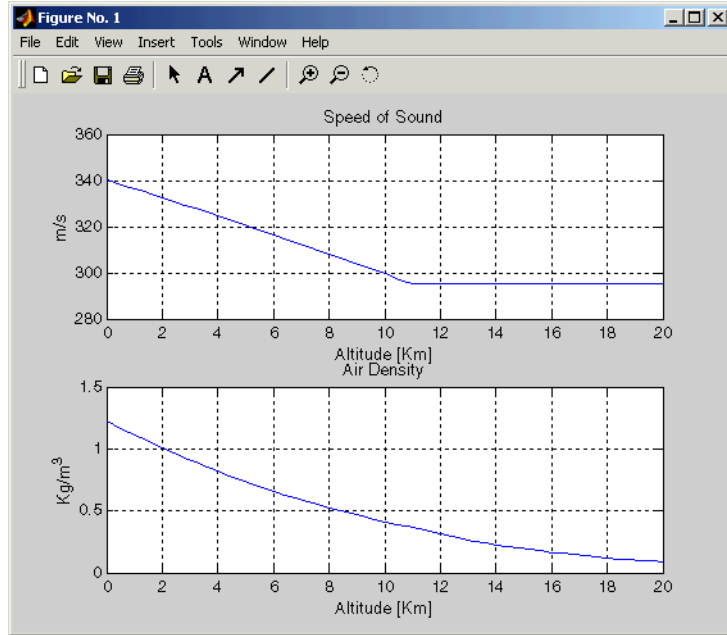
The following equations define the troposphere.

$$T = T_o - Lh$$

$$\rho = \rho_o \cdot \left(\frac{T}{T_o}\right)^{\frac{g}{LR} - 1}$$

$$P = P_o \cdot \left(\frac{T}{T_o}\right)^{\frac{g}{LR}}$$

$$a = \sqrt{\gamma R T}$$

The following equations define the lower stratosphere.

$$T = 216.7^o K$$

$$\rho \ = \ \rho_o \cdot \left(\frac{T}{T_o}\right)^{\frac{g}{LR}} \cdot e^{\frac{g}{RT}(11000 - h)}$$

$$P \ = \ P_o \cdot \left(\frac{T}{T_o}\right)^{\frac{g}{LR}-1} \cdot e^{\frac{g}{RT}(11000 - h)}$$

$$a \ = \ \sqrt{\gamma R T}$$

The symbols are defined as follows.

T_0	Absolute temperature at mean sea level in degrees Kelvin
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
T	Absolute temperature at altitude h in degrees Kelvin
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 degrees Kelvin/m
R	Characteristic gas constant J/kg-degrees Kelvin
γ	Specific heat ratio
g	Acceleration due to gravity in m/s^2

You can look under the mask of the ISA Atmosphere Model block to see how these equations are implemented in the model.

Aerodynamics & Equations of Motion Subsystem

The Aerodynamics & Equations of Motion subsystem generates the forces and moments applied to the missile in the body axes and integrates the equations of motion that define the linear and angular motion of the airframe. The aerodynamic coefficients are stored in data sets, and, during the simulation, the value at the current operating condition is determined by interpolation using the Interpolation (n-D) using PreLook-Up blocks.
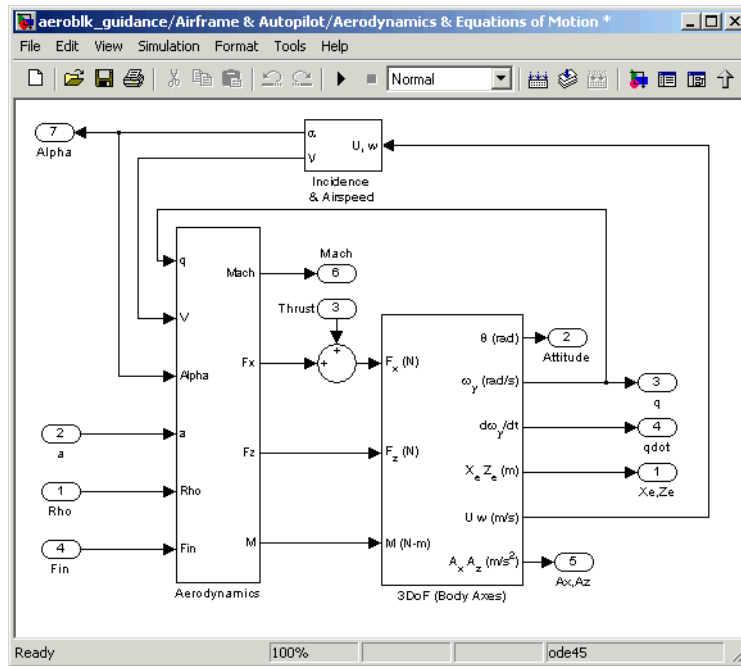
These are the three-degrees-of-freedom body axis equations of motion, which are defined in the Equations of Motion (Body Axes) block.

$$\dot{U} = (T + F_x)/m - qW - g\sin\theta$$
$$\dot{W} = F_z/m + qU + g\cos\theta$$
$$\dot{q} = M/I_{yy}$$
$$\dot{\theta} = q$$

These are the aerodynamic forces and moments equations, which are defined in the Aerodynamics subsystem.

$$F_x = \bar{q} S_{ref} C_x (Mach, \alpha)$$

$$F_z = \bar{q} S_{ref} C_z (Mach, \alpha, \eta)$$

$$M = \bar{q} S_{ref} d_{ref} C_M (Mach, \alpha, \eta, q)$$

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

These are the stability axes variables, which are calculated in the Incidence & Airspeed block.

$$V = \sqrt{U^2 + W^2}$$

$$\alpha = atan(W/U)$$

The symbols are defined as follows.

θ	Attitude in radians
q	Body rotation rate in rad/s
M	Missile mass in kg
g	Acceleration due to gravity in m/s^2
I_{yy}	Moment of inertia about the y axis in kg-m^2
$\dot{W}$	Acceleration in the Z body axis in m/s^2
$\dot{q}$	Change in body rotation rate in rad/s^2
T	Thrust in the X body axis in N
ρ	Air density in kg/m^3
S_{ref}	Reference area in m^2
C_X	Coefficient of aerodynamic force in the X axis
C_Z	Coefficient of aerodynamic force in the Z axis
C_M	Coefficient of aerodynamic moment about the Y axis

d_{ref}	Reference length in meters
η	Fin angle in radians
F_X	Aerodynamic force in the X body axis in N
F_Z	Aerodynamic force in the Z body axis in N
M	Aerodynamic moment along the Y body axis
$\bar{q}$	Dynamic pressure in Pa
V	Airspeed in m/s
α	Incidence in radians
U	Velocity in the X body axis in m/s
W	Velocity in the Z body axis in m/s

Modeling a Classical Three-Loop Autopilot

The missile autopilot controls the acceleration normal to the missile body. In this case study, the autopilot structure is a three-loop design using measurements from an accelerometer located ahead of the missile's center of gravity and from a rate gyro to provide additional damping. The following figure shows the classical configuration of an autopilot. The controller gains are scheduled on incidence and Mach number and tuned for robust performance at an altitude of 3050 meters (10000 feet).
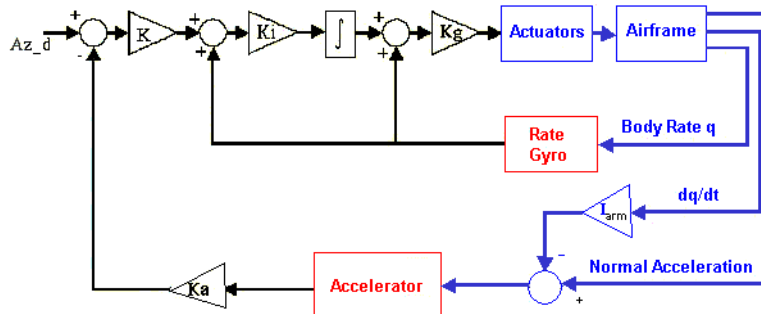
Designing an autopilot entails the following:

- "Trimming and Linearizing an Airframe Model" on page 3-11 explains how to model the airframe pitch dynamics for a number of trimmed flight conditions.
- "Autopilot Design" on page 3-12 summarizes the autopilot design process.

Trimming and Linearizing an Airframe Model

Designing the autopilot using classical design techniques requires linear models of the airframe pitch dynamics for a number of trimmed flight conditions. MATLAB can determine the trim conditions and derive linear state-space models directly from the nonlinear Simulink model. This saves time and helps to validate the model. The functions provided by the Control System Toolbox allow you to visualize the behavior of the airframe in terms of open-loop frequency (or time) responses.

The airframe trim demo shows how to trim and linearize an airframe model. To run this demo, enter the following in the MATLAB Command Window.

```
aeroblk_lin_aero
```

The output from this demo is a Bode diagram in the Control System Toolbox viewer.
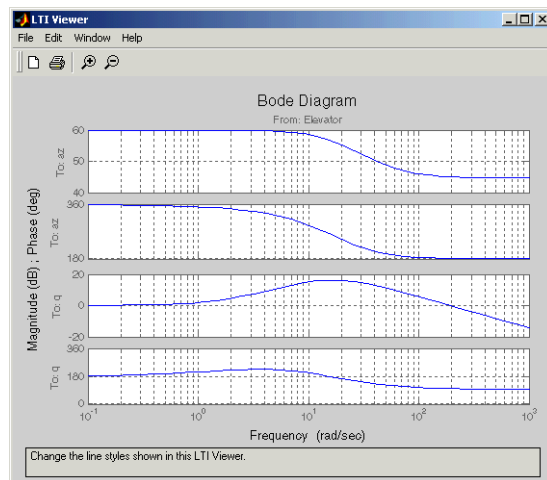
Autopilot Design

Autopilot design can begin after the missile airframe has been linearized at a number of flight conditions. Typically, autopilot designs are carried out on a number of linear airframe models derived at varying flight conditions across the expected flight envelope. Implementing the autopilot in the nonlinear model involves storing the autopilot gains in two-dimensional lookup tables and incorporating an antiwindup gain to prevent integrator windup when the fin demands exceed the maximum limits. Testing the autopilot in the nonlinear Simulink model is the best way to demonstrate satisfactory performance in the presence of nonlinearities, such as actuator fin and rate limits and dynamically changing gains.

The Autopilot subsystem is an implementation of the classical three-loop autopilot design within Simulink.
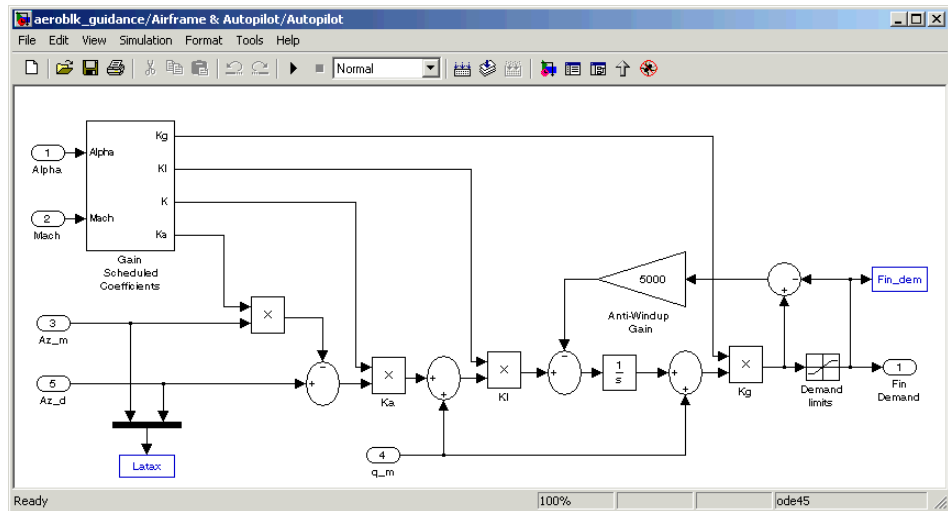
Modeling the Homing Guidance Loop

The complete homing guidance loop consists of these two subsystems:

The autopilot is now part of an inner loop within the overall homing guidance system. Consult Reference [4] for information on different types of guidance systems and on the analysis techniques that are used to quantify guidance loop performance.
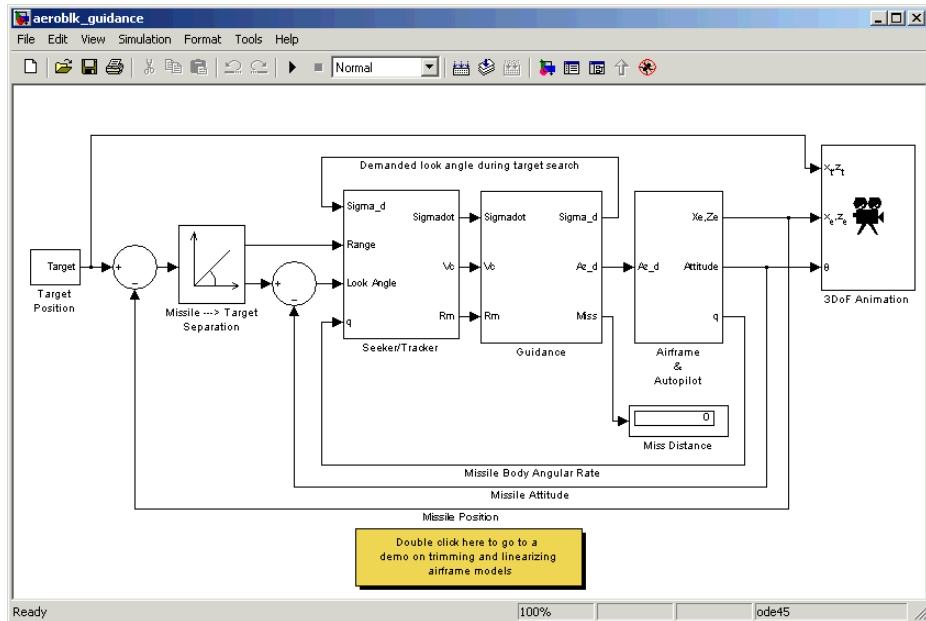
Guidance Subsystem

Initially, the Guidance subsystem searches to locate the target's position and then generates demands during closed-loop tracking. A Stateflow model controls the transfer between the different modes of these operations. Stateflow is the ideal tool for rapidly defining all the operational modes, both during normal operation and during unusual situations.
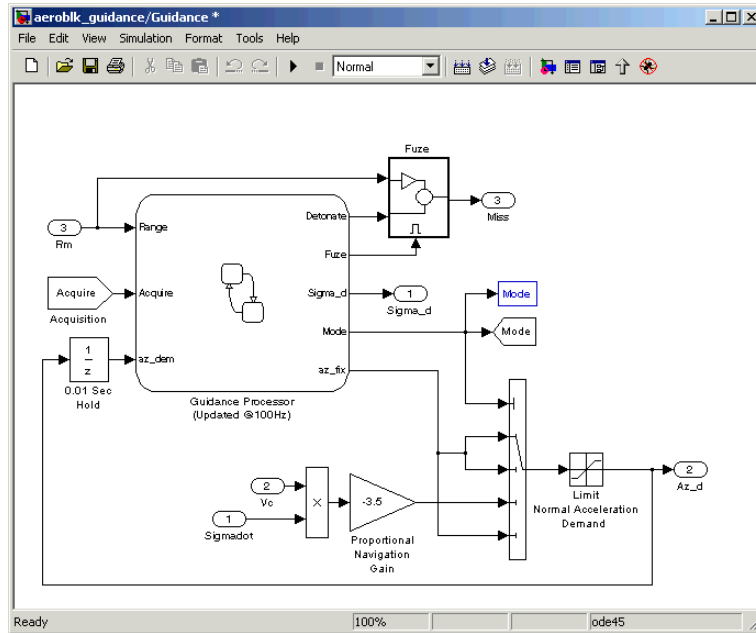
Guidance Processor Statechart. Mode switching is triggered by events generated in Simulink or in the Stateflow chart. The variable Mode is passed out to Simulink and is used to control the Simulink model's behavior and to determine the response of the Simulink model. For example, the Guidance Processor state chart, which is part of the Guidance subsystem, shows how the system reacts in response to either losing the target lock or failing to acquire the target's position during the target search.

During the target search, this Stateflow state chart controls the tracker directly by sending demands to the seeker gimbals (Sigma_d). Target acquisition is flagged by the tracker once the target lies within the beam width of the seeker (Acquire) and, after a short delay, closed loop guidance begins.
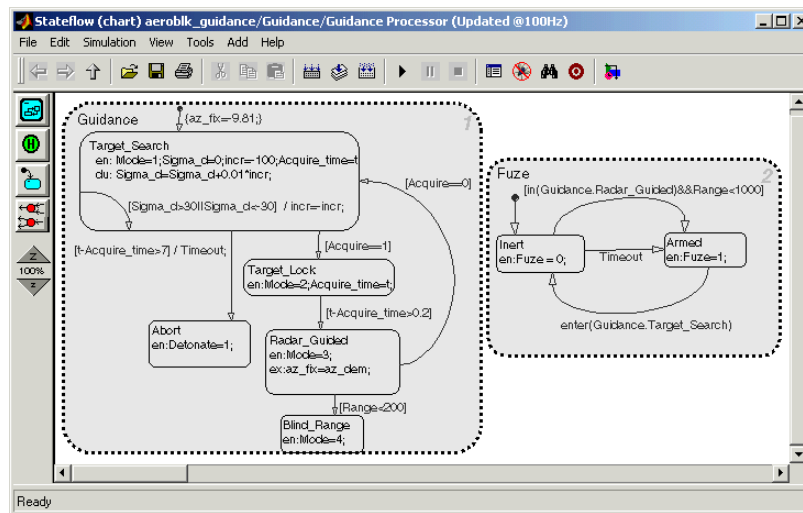
Proportional Navigation Guidance Measurements. Once the seeker has acquired the target, a Proportional Navigation Guidance (PNG) law guides the missile until impact. This form of guidance law is the most basic, used in guided missiles since the 1950s, and can be applied to radar-, infrared-, or television-guided missiles. The navigation law requires measurements of the closing velocity between the missile and target, which for a radar-guided missile can be obtained with a Doppler tracking device, and an estimate for the rate of change of the inertial sight line angle.

Proportional Navigation Guidance Measurements

3-15
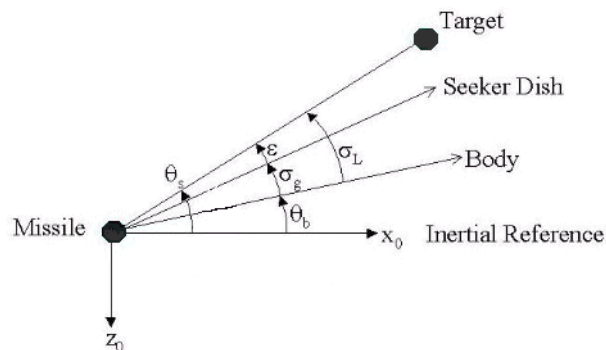
The diagram symbols are defined as follows.

λ	Navigation gain (> 2)
V_c	Closing velocity
θ_b	Body attitude
$\dot{\theta}_s$	Sight line rate
σ_g	Gimbal angle
σ_L	Look angle
σ_d	Dish angle
$a_{z\_dem} = \lambda V_c \dot{\theta}_s$	Demanded normal acceleration

Seeker/Tracker Subsystem

The Seeker/Tracker subsystem controls the seeker gimbals to keep the seeker dish aligned with the target and provides the guidance law with an estimate of the sight line rate.
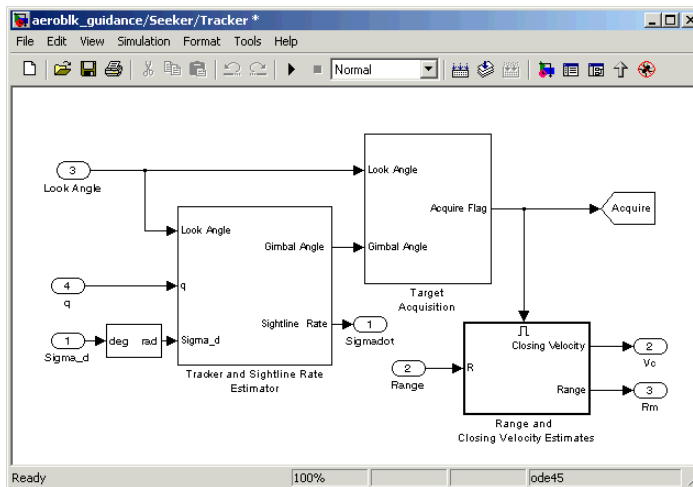
Tracker and Sightline Rate Estimator. The Tracker and Sightline Rate Estimator, the most elaborate subsystem of the Seeker/Tracker subsystem because of its complex error modeling, is shown below.

The subsystem contains a number of feedback loops, estimated parameters, and parasitic effects for the homing guidance. The tracker loop time constant `tors` is set to `0.05` second, a compromise between maximizing speed of response and keeping the noise transmission within acceptable levels. The stabilization loop compensates for body rotation rates, and the gain `Ks`, which is the loop crossover frequency, is set as high as possible subject to the limitations of the stabilizing rate gyro's bandwidth. The sight line rate estimate is a filtered value of the sum of the rate of change of the dish angle measured by the stabilizing rate gyro and an estimated value for the rate of change of the angular tracking error (`e`) measured by the receiver. In this demo, the bandwidth of the estimator filter is set to half that of the bandwidth of the autopilot.
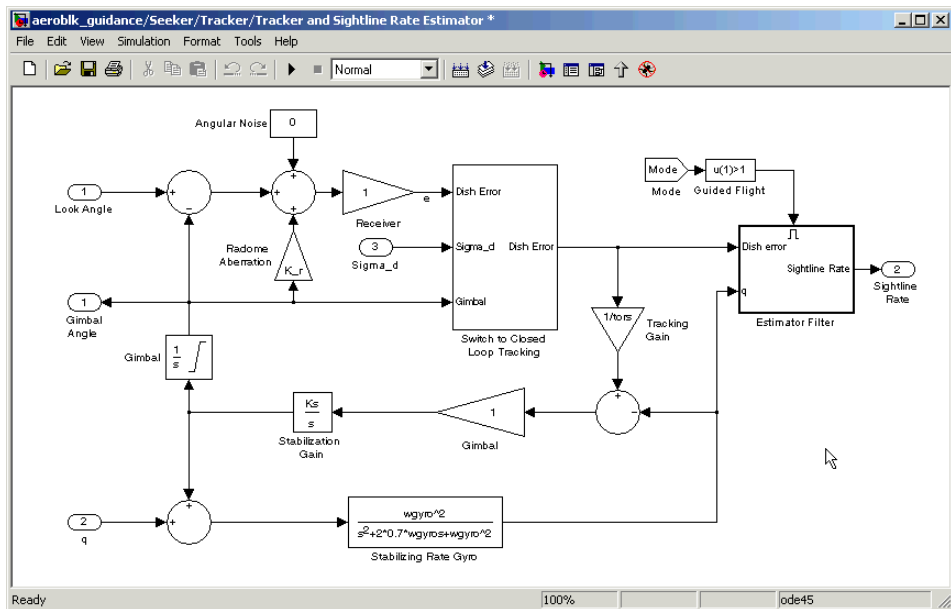
Radome Aberration. Radome aberration is also modeled by the Tracker and Sightline Rate Estimator subsystem.

Radome aberration is a parasitic feedback effect commonly modeled in radar-guided missile designs. It occurs because the shape of the protective covering over the seeker distorts the returning signal, and it gives a false reading of the look angle to the target. The amount of distortion is, in general,

a nonlinear function of the current gimbal angle. But a commonly used approximation is to assume a linear relationship between the gimbal angle and the magnitude of the distortion. Often, other parasitic effects, such as sensitivity to normal acceleration in the rate gyros, are also modeled to test the robustness of the target tracker and estimator filters.
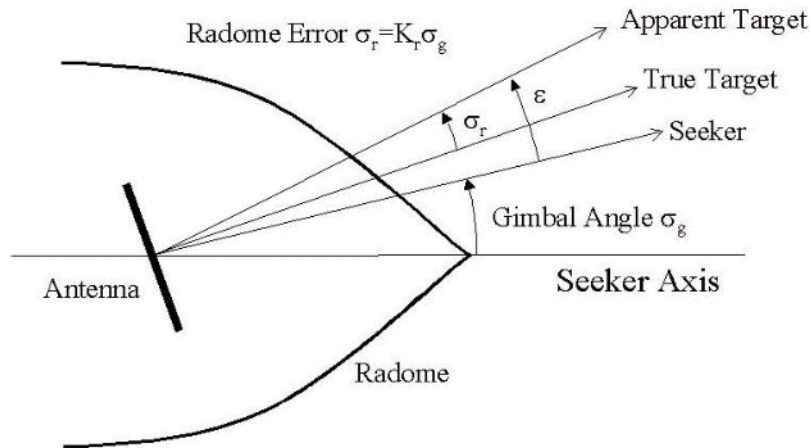
Simulating the Missile Guidance System

Running the guidance simulation demonstrates the performance of the overall system. The target is defined to be traveling at a constant speed of 328 m/s on a reciprocal course to the initial missile heading and 500 meters above the initial missile position. The data, shown in the figure below, can be used to determine if the missile can withstand the flight demands and complete the mission to impact.
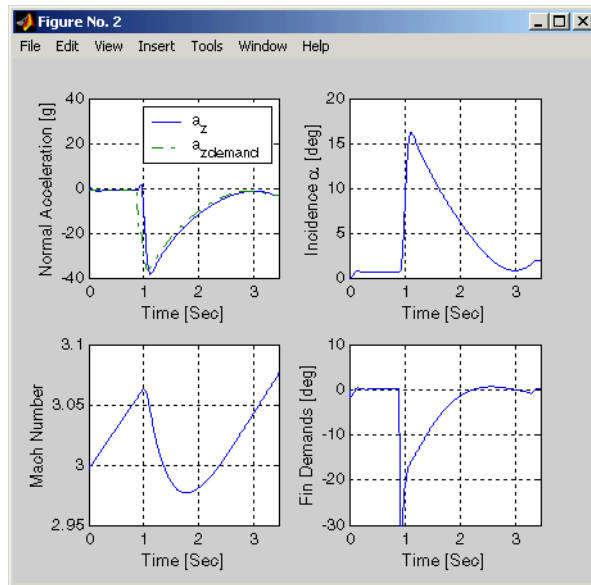
The simulation results show that target acquisition occurs 0.69 second after search initiation, with closed loop guidance starting after 0.89 second. Impact with the target occurs at 3.46 seconds, with the range to target at the point of closest approach calculated to be 0.26 meter.
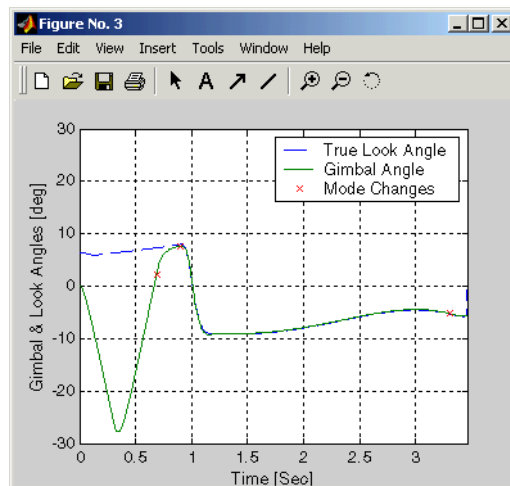
Extending the Model

Modeling the airframe and guidance loop in a single plane is only the start of the design process. Extending the model to a full six-degrees-of-freedom representation requires the implementation of the full equations of motion for a rigid body.

Six-degrees-of-freedom can be represented using quaternion or Euler Angles. The quaternion implementation uses a quaternion to represent the angular orientation of the body in space. The quaternion is appropriate when the standard Euler angle definitions become singular as the pitch attitude tends to ±90 degrees. The Euler angle implementation uses the standard Euler angle equations of motion. Euler angles are appropriate when obtaining trim conditions and modeling linear airframes. This model contains one of the six-degrees-of-freedom equations of motion blocks.
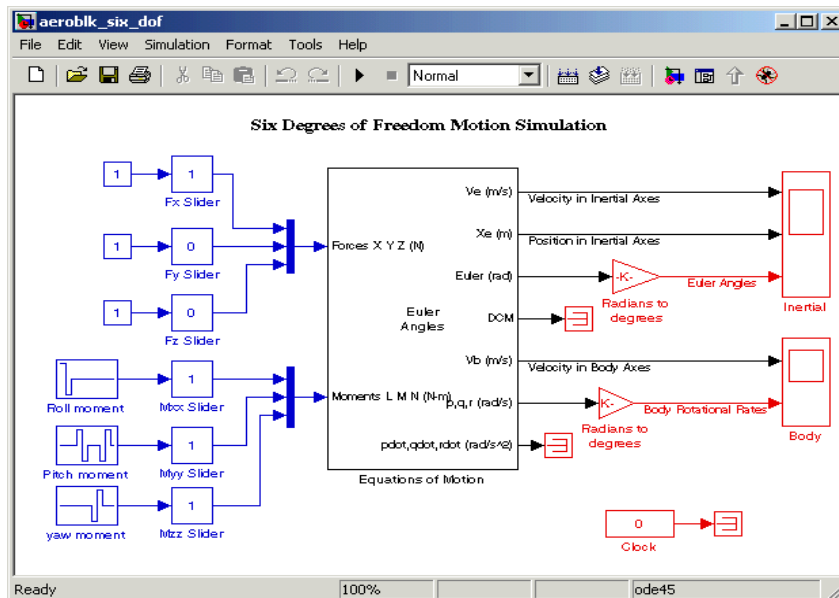
References

[1] Bennani, S., D. M. C. Willemsen, and C. W. Scherer, "Robust LPV control with bounded parameter rates," AIAA-97-3641, August 1997.

[2] Mracek, C. P. and J. R. Cloutier, "Full Envelope Missile Longitudinal Autopilot Design Using the State-Dependent Riccati Equation Method," AIAA-97-3767, August 1997.

[3] Shamma, J. S. and J. R. Cloutier, "Gain-Scheduled Missile Autopilot Design Using Linear Parameter Varying Transformations," *Journal of Guidance, Control and Dynamics*, Vol. 16, No. 2, March-April 1993.

[4] Lin, Ching-Fang, *Modern Navigation, Guidance, and Control Processing*, Vol. 2, ISBN 0-13-596230-7, Prentice Hall, 1991.

NASA HL-20 Lifting Body Airframe

This section shows how to model the airframe of a NASA HL-20 lifting body, a low-cost complement to the Space Shuttle orbiter, with Simulink and the Aerospace Blockset.

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 case study, the Aerospace Blockset is used to rapidly model portions of the HL-20 airframe. The remaining portions, including the calculation of the aerodynamic coefficients, are modeled with Simulink. This case study examines the construction of the Simulink model of the HL-20 airframe and touches on how the aerodynamic data are used in the model.

This section includes the following topics:

- "NASA HL-20 Lifting Body" on page 3-22 provides an overview of the history and purposes of the NASA HL-20 lifting body.
- "The HL-20 Airframe Model" on page 3-24 describes how the Aerospace Blockset and Simulink are used to model the HL-20 airframe.
- "References" on page 3-35 provides a selected bibliography.

NASA HL-20 Lifting Body

The HL-20, also known as 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 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 an 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 runways.

Potential uses for the HL-20 include

- Orbital rescue of stranded astronauts
- International Space Station crew exchanges, if the Space Shuttle orbiter is not available
- 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 Model

You can open the HL20 airframe model by entering `aeroblk_HL20_main` at the command line.
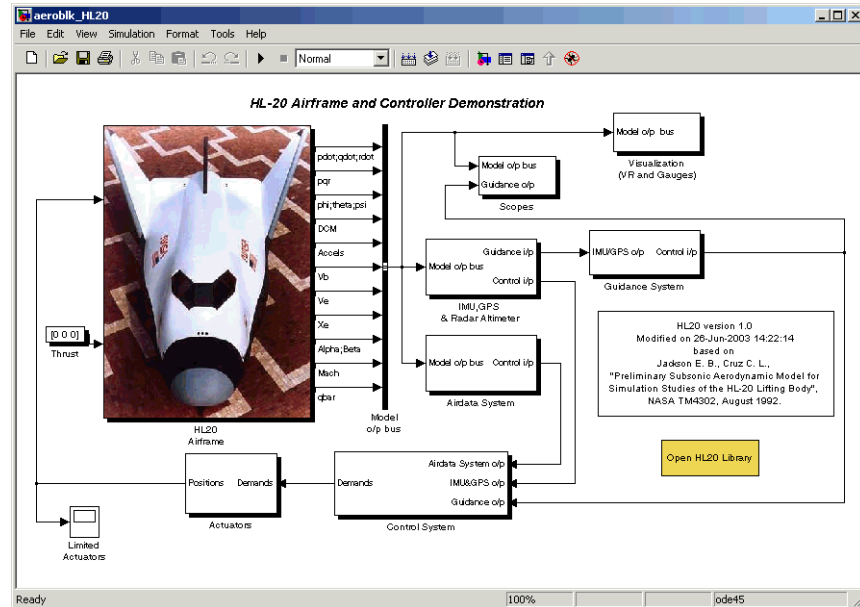
HL-20 Airframe Model

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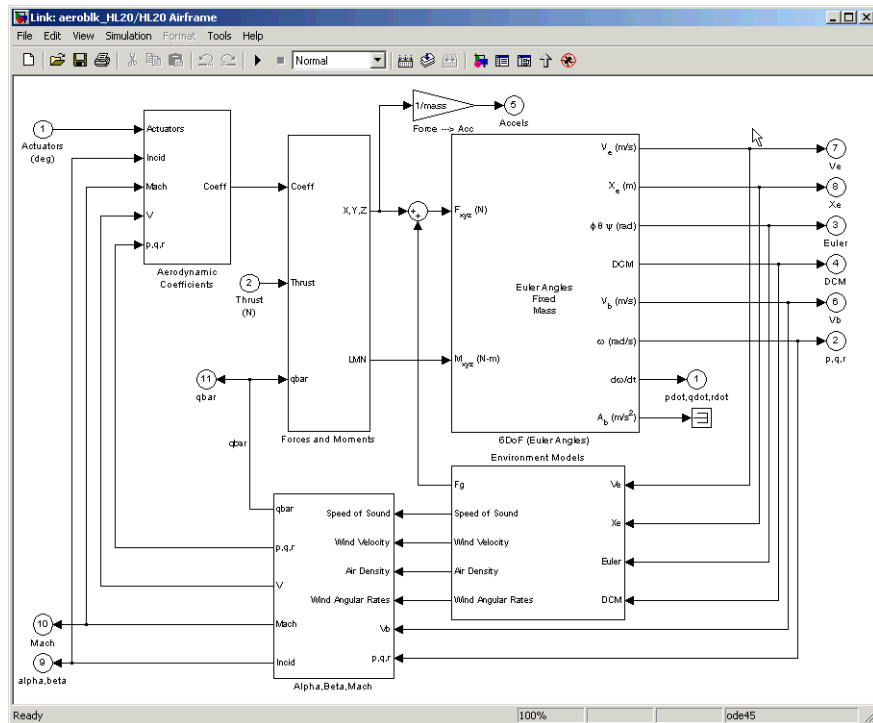
HL-20 Airframe Subsystem

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. 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 blocks/subsystems:

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

These are some of the standard environmental blocks contained in the Aerospace Blockset. The environmental models implement mathematical representations within standard references, such as U.S. Standard Atmosphere, 1976.

The following figures show the environmental and wind turbulence models used in the model.
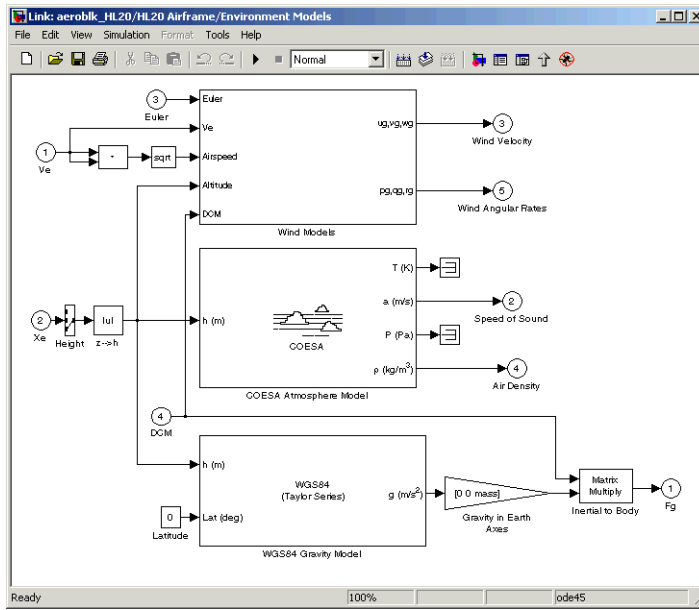
Environmental Models in HL-20 Airframe Model
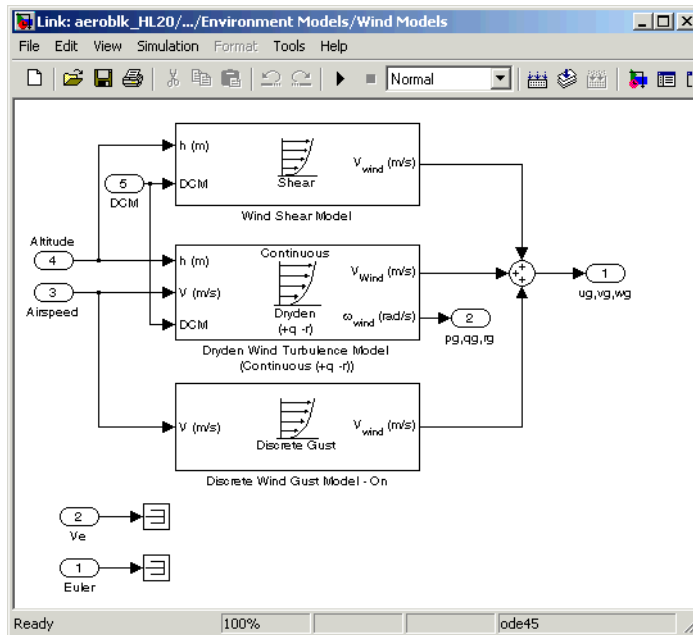
Wind Models in HL-20 Airframe Model

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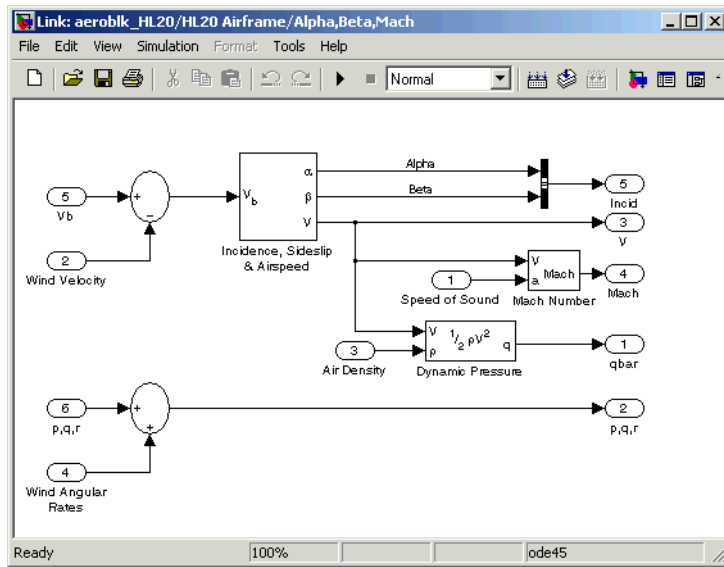
Additional Computed Parameters for HL-20 Airframe Model (Alpha, Beta, Mach Subsystem)

Aerodynamic Coefficients Subsystem

The Aerodynamic Coefficients subsystem contains aerodynamic data and equations for calculating the six aerodynamic coefficients, which are implemented as in NASA document TM4302. However, the ground and landing gear effects are not used in this aerodynamic model. The six aerodynamic coefficients are as follows.

Cx	Axial-force coefficient
Cy	Side-force coefficient
Cz	Normal-force coefficient
Cl	Rolling-moment coefficient
Cm	Pitching-moment coefficient
Cn	Yawing-moment coefficient

The contribution of each of these is calculated in the subsystems (body rate, actuator increment, and datum), and then summed and passed to the Forces and Moments subsystem.
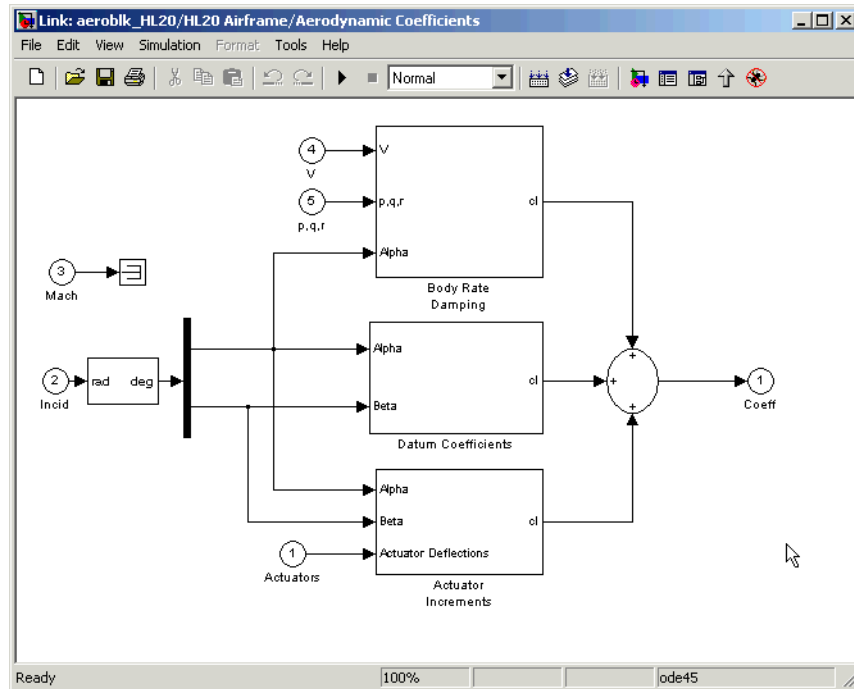
Aerodynamic Coefficients in HL-20 Airframe Model

Aerodynamic Coefficient Calculation. 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 NASA document TM4302 [1].

The polynomial functions contained in the M-file aeroblk_init_hl20.m 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

model efficiency, most tables are implemented as PreLook-up Index Search and Interpolation (n-D) using PreLook-up blocks. These blocks improve efficiency most when there are 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 3-32
- "Body Rate Damping Subsystem" on page 3-33
- "Actuator Increment Subsystem" on page 3-33

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.

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.
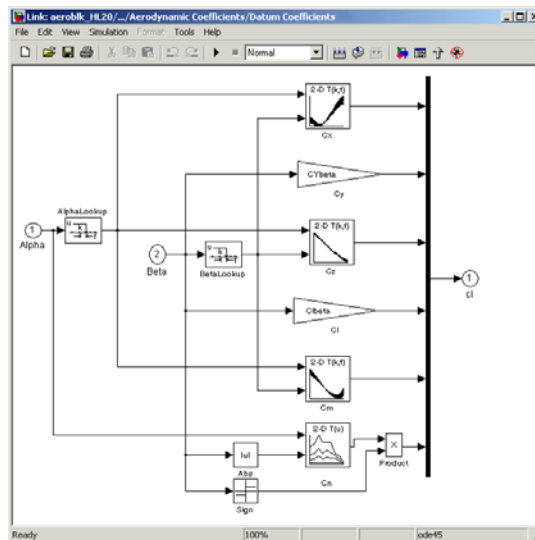
Datum Coefficients Subsystem

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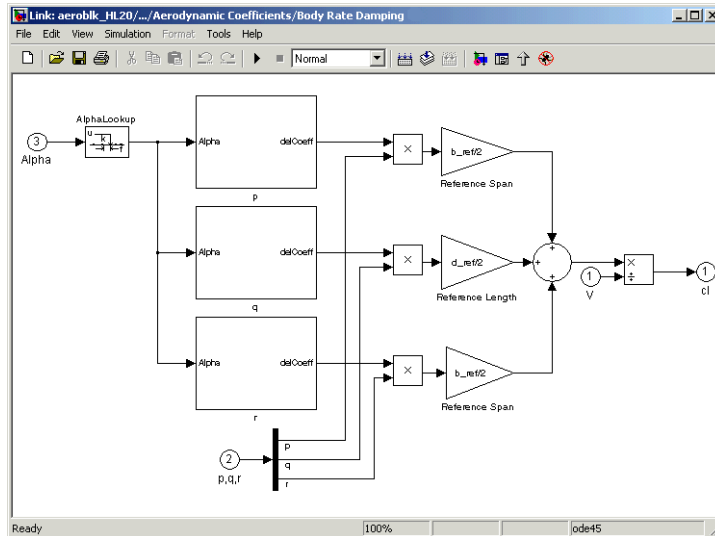
Body Rate Damping Subsystem

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.
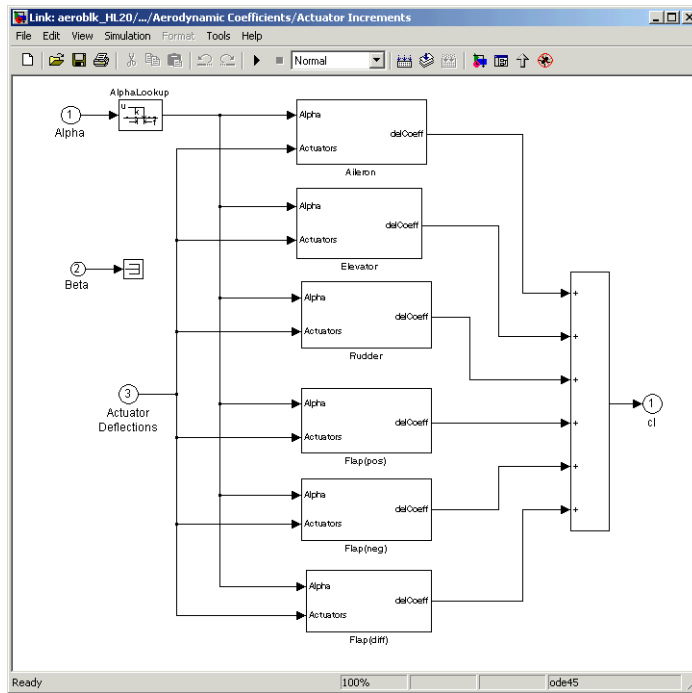
Actuator Increments Subsystem

Forces and Moments Subsystem. The last subsystem in the HL-20 airframe model is Forces and Moments. 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. The equations defining the body forces and body moments are found in NASA document TM4302 [1].

Forces and Moments Subsystem

Completing the Model

The Simulink and the Aerospace Blockset 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 by entering `aeroblk_HL20_main` in the command window.

References

Additional information about the HL-20 lifting body can be found at `http://www.astronautix.com/craft/hl20.htm`.

[1] Jackson E. B., and C. L. Cruz, C. L., "Preliminary Subsonic Aerodynamic Model for Simulation Studies of the HL-20 Lifting Body," NASA TM4302 (August 1992). This document is included in the zip file available from MATLAB Central.

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

See also:
`http://www.aviationnow.com/content/publication/awst/20020520/aw46.htm`.

Ideal Airspeed Correction

This case study demonstrates how to create indicated and true airspeed using Simulink and the Aerospace Blockset. To find out more, read the following sections:

- "Airspeed Correction Models" on page 3-36 shows how to open the models that are used in this case study.
- "Measuring Airspeed" on page 3-37 describes the different types of airspeed used in aerospace engineering.
- "Modeling Airspeed Correction" on page 3-38 describes how the Ideal Airspeed Correction block is implemented in the two models.
- "Simulating Airspeed Correction" on page 3-41 describes how to run the model simulation.

Airspeed Correction Models

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

```
aeroblk_indicated
```

and

```
aeroblk_calibrated
```
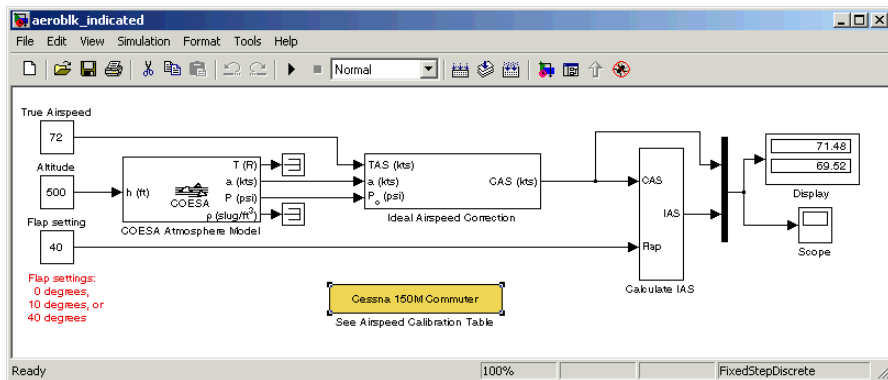

aeroblk_indicated Model
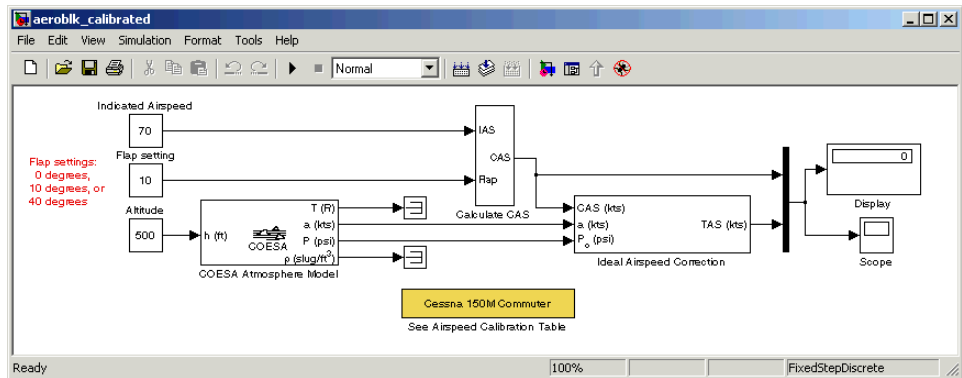
aeroblk_calibrated Model

Measuring Airspeed

To measure airspeed, most light aircraft designs implement pilot-static airspeed indicators. Pilot-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), which denotes the airspeed that a pilot would see in the cockpit of an aircraft.

To help compensate for measurement errors, airspeed is divided into three definitions of measurement.

Airspeed Type	Description	See Also
Calibrated	Indicated airspeed that is corrected for the calibration error	"Examining the Calibration Error" on page 3-38
Equivalent	Calibrated airspeed that is corrected for the compressibility error	"Examining the Compressibility Error" on page 3-38
True	Equivalent airspeed that is corrected for the density error	"Examining the Density Error" on page 3-38

Examining the Calibration Error

An airspeed indicator features a static vent to maintain a pressure equal to atmospheric pressure inside the instrument. Position and placement of the static vent along with angle of attack and velocity of the aircraft will determine the pressure inside the airspeed indicator, and thereby, the amount of calibration error of the airspeed indicator. Therefore, 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.

Examining the Compressibility Error

The ability of air to resist compression diminishes as altitude and airspeed increases, or when contained in a restricted volume. A restricted volume of air exists within a pilot-static airspeed indicator. When flying at high altitudes and high airspeeds, calibrated airspeed is always higher than equivalent airspeed. Equivalent airspeed can be derived by compensating the calibrated airspeed for the compressibility error.

Examining the Density Error

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

Modeling 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 airplane. The `aeroblk_indicated` model implements a conversion to indicated airspeed, and the `aeroblk_calibrated` model implements a conversion to true airspeed.

Each model consists of two main components:

- "COESA Atmosphere Model Block" on page 3-39 calculates the change in atmospheric conditions with changing altitude.
- "Ideal Airspeed Correction Block" on page 3-39 transforms true airspeed to calibrated airspeed and vice versa.

COESA Atmosphere Model Block

The COESA Atmosphere Model block is a 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 32000 meters (approximately 104987 feet), the U.S. Standard Atmosphere is identical with the Standard Atmosphere of the International Civil Aviation Organization (ICAO).

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 lets you compensate for the airspeed measurement errors to convert airspeed from one type to another type. The following table contains the Ideal Airspeed Correction block's airspeed 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 airspeed to calibrated airspeed. In the `aeroblk_calibrated` model, the Ideal Airspeed Correction block transforms calibrated airspeed to true airspeed.

To understand how the Ideal Airspeed Correction block implements airspeed transformations as mathematical formulas, see the following sections:

True Airspeed Implementation. True airspeed (TAS) is implemented as an input and as a function of equivalent airspeed (EAS), which can be expressed as

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

The symbols are defined as follows.

a	Speed of sound at altitude in m/s^2
δ	Relative pressure ratio at altitude
a_0	Speed of sound at mean sea level in m/s^2

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

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

The symbols are defined as follows:

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

Equivalent Airspeed Implementation. Equivalent airspeed (EAS). which is derived using the compressible form of Bernoulli's equation and assuming isentropic conditions, can be expressed as:

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

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

Simulating 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 an 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.

As you can see in the following figure, the display shows both indicated airspeed and calibrated airspeed.
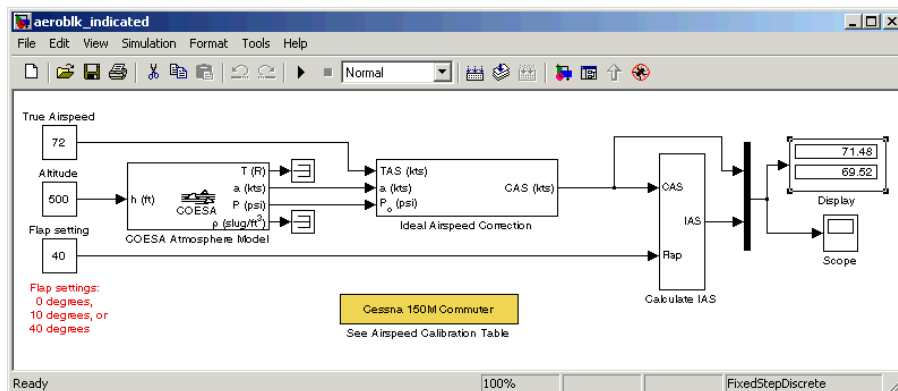
In the `aeroblk_calibrated` model, the aircraft is defined to be traveling at a constant speed of 70 knots (indicated airspeed) and altitude of 500 feet. The flaps are set to 10 degrees. The COESA Atmosphere Model block takes the altitude as an input and outputs the speed of sound and air pressure. The `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.

As you can see in the following figure, the display shows both calibrated airspeed and true airspeed.
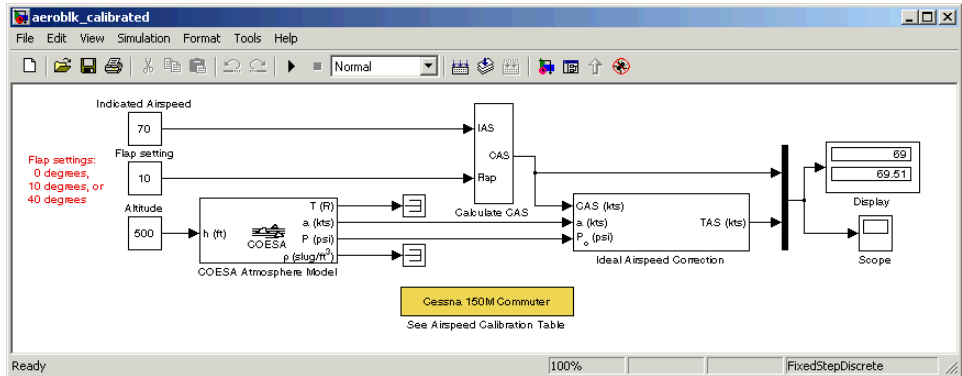
1903 Wright Flyer Model

This section describes the 1903 Wright Flyer model, which was built using Simulink, the Aerospace Blockset, and the Virtual Reality Toolbox. Built by Orville and Wilbur Wright, the Wright Flyer took to the skies in December 1903 and represented the dawn of the aviation age. 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 created by Frederick J. Hooven[1]. That simulation explored the longitudinal stability of the Wright Flyer, and therefore, dealt only with the 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 over turn in crosswinds and gusts, and longitudinally, the 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.

For more information, see the following sections:

• "Wright Flyer Model" on page 3-44 shows how to open the model used in this case study.

• "Airframe Subsystem" on page 3-44 describes the airframe subsystem.

• "Environment Subsystem" on page 3-49 describes the environment subsystem.

• "Pilot Subsystem" on page 3-51 describes the pilot subsystem.

• "Running the Simulation" on page 3-51 provides a demonstration of the Wright Flyer model.

Wright Flyer Model

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

```
aeroblk_wf_3dof
```


aeroblk_wf_3dof Model

Airframe Subsystem

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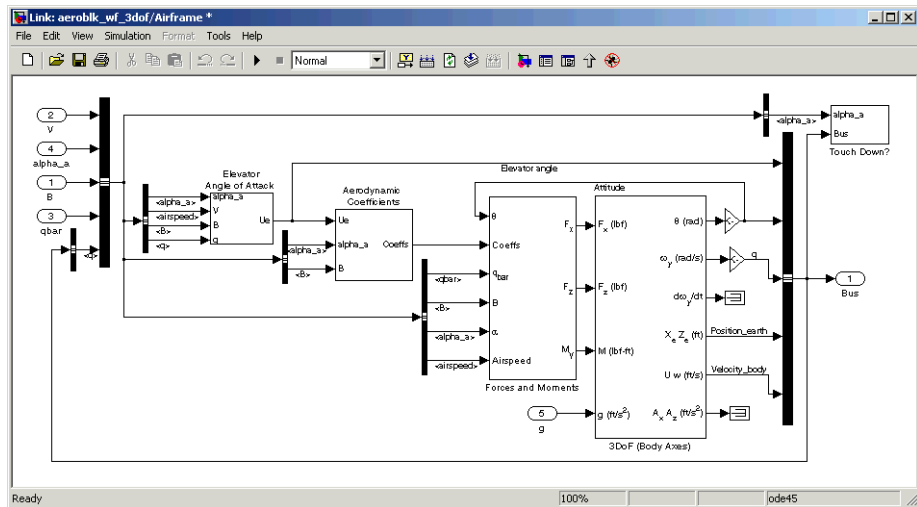
Airframe Subsystem

The Airframe subsystem consists of the following parts:

- "Elevator Angle of Attack Subsystem" on page 3-45
- "Aerodynamic Coefficients Subsystem" on page 3-46
- "Forces and Moments Subsystem" on page 3-47
- "3DoF (Body Axes) Block" on page 3-48

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.
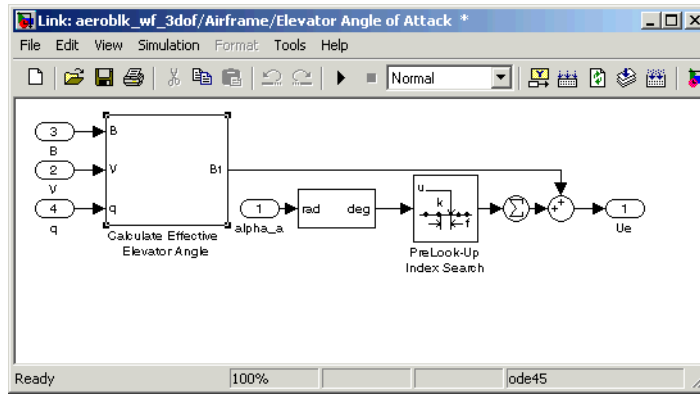
Elevator Angle of Attack Subsystem

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 PreLook-Up blocks.
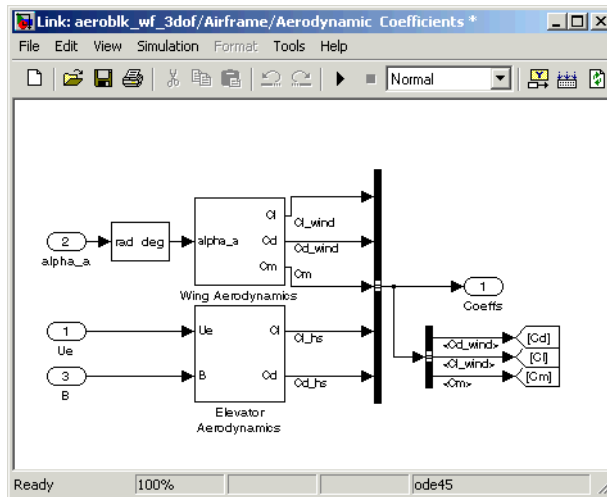
Aerodynamic Coefficients Subsystem

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.
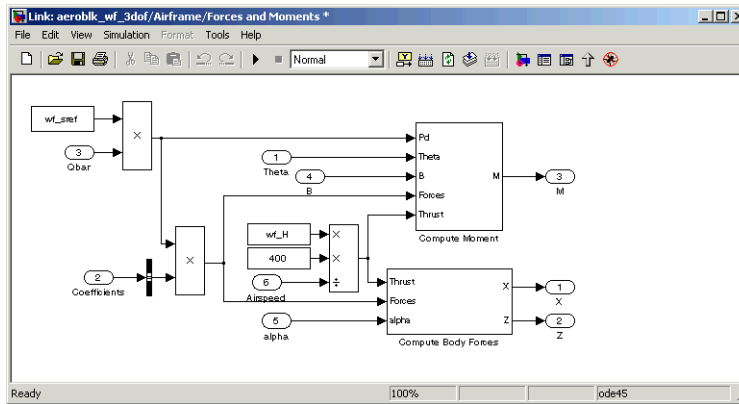
Forces and Moments Subsystem

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. Conversions from the axis system of Hooven's simulation and body axes are also performed.
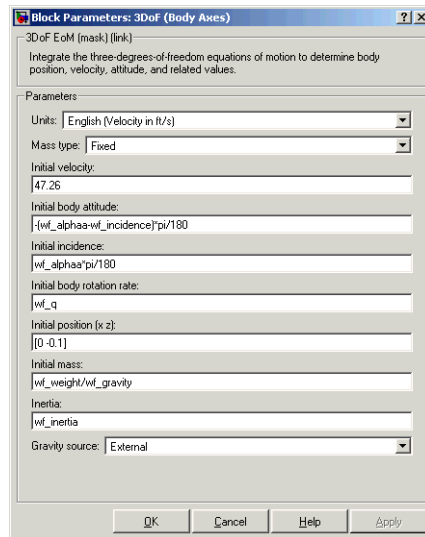
3DoF (Body Axes) Block Mask

Environment Subsystem

The Environment subsystem of the Wright Flyer model contains a variety of blocks from the Environment sublibrary of the Aerospace Blockset, including wind, atmosphere, and gravity. In addition, the Environment subsystem calculates airspeed and dynamic pressure.
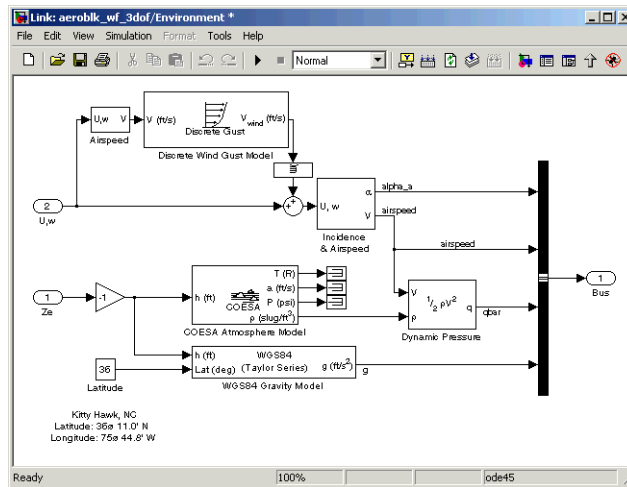
Environment Subsystem

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, which is situated near the Atlantic coast. On that day, wind gusts of more than 25 m.p.h. were recorded at Kitty Hawk. Indeed, after the final flight on that blustery December day, a wind gust caught the Wright Flyer and flipped it over, damaging it beyond repair.

In the Wright Flyer model's Environmental subsystem, the Discrete Wind Gust Model block provides wind gusts to the simulated environment. The other blocks are

- The Incidence and Airspeed block calculates the angle of attack and airspeed.
- The COESA Atmosphere Model block calculates the air density.
- The Dynamic Pressure block takes the air density and velocity and computes the dynamic pressure.
- The WGS84 Gravity Block produces the gravity at the Wright Flyer's latitude 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 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 contributes to the success of the flights. Hooven found the reaction time of 0.06 second to be optimum for successful flights. The Delay of Pilot block recreates this effect by producing a delay of no more than 0.08 second.

Pilot Subsystem

Running the Simulation

Default values for this simulation are set to allow the Wright Flyer model to take off and land successfully. The pilot reaction time (`wf_B3`) is set to 0.06 second, 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. Examining the attitude, this model exhibits the longitudinal "undulation" of the original aircraft.

Attitude Scope

A pilot with quick reaction times and ideal flight conditions make it possible to fly the Wright Flyer successfully. The Wright Flyer model confirms that it was quite challenging to control the longitudinal motion of the Flyer. The longest recorded flight on that day lasted a mere 59 seconds and covered 852 feet.

Using the Virtual Reality Toolbox, the Wright Flyer model also provides a Virtual Reality Modeling Language (VRML) world. Thaddeus Beier created the initial Wright Flyer model in Inventor format.

The VR Sink block enables the flight motion to be viewed in three dimensions.
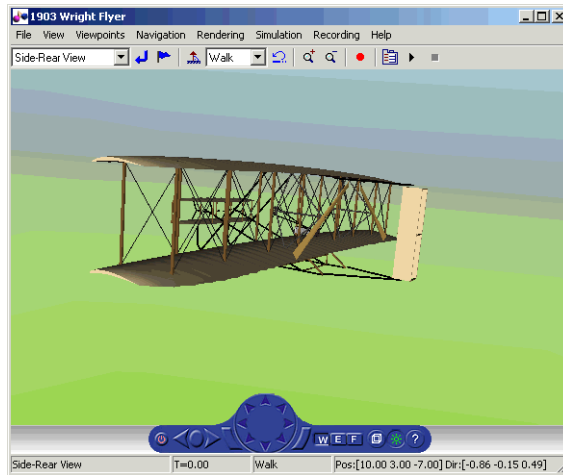
1903 Wright Flyer Virtual Reality World

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* edited by Howard S. Wolko, 1987.

[2] Culick, F. E. C. and Jex, H. R., "Aerodynamics, Stability, and Control of the 1903 Wright Flyer," from *The Wright Flyer An Engineering Perspective* edited by Howard S. Wolko, 1987.

Additional information about the 1903 Wright Flyer can be found at the following locations:

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

4

Block Reference

Blocks — Categorical List

The Aerospace Blockset's block library, `aerolib`, is organized into libraries according to their behavior. The **aerolib** window displays the block library icons and names.

Actuators Library	Actuator models
Aerodynamics Library	Aerodynamics models
Animation Library	3-D animation during simulation
Environment Library	Environmental models, including the Atmosphere Sublibrary, the Gravity Sublibrary, and the Wind Sublibrary
Flight Parameters Library	Flight parameter models
Equations of Motion Library	Equation of motion models, including the 3DoF Sublibrary and the 6DoF Sublibrary
GNC Library	Gain scheduling models, including the Controls Sublibrary and the Guidance Sublibrary
Mass Properties Library	Center of gravity and tensor models
Propulsion Library	Simple propulsion system models
Utilities Library	Common mathematical operations and conversions, including the Axes Transformations Sublibrary, the Unit Conversions Sublibrary, and the Math Operations Sublibrary

Actuators Library

Second Order Linear
Actuator

Implement a second-order linear actuator

Second Order Nonlinear
Actuator

Implement a second-order nonlinear actuator
with rate and deflection limits

Aerodynamics Library

Aerodynamic Forces and
Moments

Compute the aerodynamic forces and moments
using the aerodynamic coefficients, dynamic
pressure, center of gravity, and center of
pressure

Animation Library

3DoF Animation

Create a 3-D Handle Graphics® animation of a
three-degrees-of-freedom object

6DoF Animation

Create a 3-D Handle Graphics animation of a
six-degrees-of-freedom object

Environment Library

The Environment Library contains the following sublibraries:

Atmosphere Sublibrary

COESA Atmosphere Model	Implement the 1976 Committee on Extension to the Standard Atmosphere (COESA) lower atmosphere
ISA Atmosphere Model	Implement the International Standard Atmosphere (ISA)
Lapse Rate Model	Implement Lapse Rate Model for atmosphere
Non-Standard Day 210C	Implement the MIL-STD-210C climatic data
Non-Standard Day 310	Implement the MIL-HDBK-310 climatic data
Pressure Altitude	Calculate pressure altitude based on ambient pressure

Gravity Sublibrary

WGS84 Gravity Model	Implement the 1984 World Geodetic System representation of Earth's gravity
World Magnetic Model 2000	Calculate the Earth's magnetic field at a specific location and time using the World Magnetic Model 2000 (WMM2000)

Wind Sublibrary

Discrete Wind Gust Model	Generate discrete wind gust
Dryden Wind Turbulence Model (Continuous)	Generate wind turbulence with the Dryden velocity spectra
Dryden Wind Turbulence Model (Discrete)	Generate wind turbulence with the Dryden velocity spectra
Horizontal Wind Model	Transform horizontal wind into body-axes coordinates
Von Karman Wind Turbulence Model (Continuous)	Generate atmospheric turbulence
Wind Shear Model	Calculate wind shear conditions

Flight Parameters Library

Dynamic Pressure	Compute dynamic pressure using velocity and air density
Ideal Airspeed Correction	Calculate equivalent airspeed (EAS), calibrated airspeed (CAS), or true airspeed (TAS) from each other
Incidence & Airspeed	Calculate incidence and air speed
Incidence, Sideslip & Airspeed	Calculate incidence, sideslip and air speed
Mach Number	Compute Mach number using velocity and speed of sound
Relative Ratio	Calculate relative atmospheric ratios

Equations of Motion Library

The Equations of Motion library contains the following sublibraries:

3DoF Sublibrary

3DoF (Body Axes)	Implement three-degrees-of-freedom equations of motion
Custom Variable Mass 3DoF (Body Axes)	Implement three-degrees-of-freedom equations of motion
Simple Variable Mass 3DoF (Body Axes)	Implement three-degrees-of-freedom equations of motion

6DoF Sublibrary

6DoF (Euler Angles)	Implement an Euler angle representation of six-degrees-of-freedom equations of motion
6DoF (Quaternion)	Implement a quaternion representation of six-degrees-of-freedom equations of motion
Custom Variable Mass 6DoF (Euler Angles)	Implement an Euler angle representation of six-degrees-of-freedom equations of motion
Custom Variable Mass 6DoF (Quaternion)	Implement a quaternion representation of six-degrees-of-freedom equations of motion
Simple Variable Mass 6DoF (Euler Angles)	Implement an Euler angle representation of six-degrees-of-freedom equations of motion
Custom Variable Mass 6DoF (Quaternion)	Implement a quaternion representation of six-degrees-of-freedom equations of motion

GNC Library

The GNC library contains the following sublibraries:

Controls Sublibrary

1D Controller [A(v),B(v),C(v),D(v)]	Implement a gain-scheduled state-space controller depending on one scheduling parameter
1D Controller Blend u=(1-L).K1.y+L.K2.y	Implement a 1-D vector of state-space controllers by linear interpolation of their outputs
1D Observer Form [A(v),B(v),C(v),F(v),H(v)]	Implement a gain-scheduled state-space controller in an observer form depending on one scheduling parameter
1D Self-Conditioned [A(v),B(v),C(v),D(v)]	Implement a gain-scheduled state-space controller in a self-conditioned form
2D Controller [A(v),B(v),C(v),D(v)]	Implement a gain-scheduled state-space controller depending on two scheduling parameters
2D Controller Blend	Implement a 2-D vector of state-space controllers by linear interpolation of their outputs
2D Observer Form [A(v),B(v),C(v),F(v),H(v)]	Implement a gain-scheduled state-space controller in an observer form depending on two scheduling parameters
2D Self-Conditioned [A(v),B(v),C(v),D(v)]	Implement a gain-scheduled state-space controller in a self-conditioned form
3D Controller [A(v),B(v),C(v),D(v)]	Implement a gain-scheduled state-space controller depending on three scheduling parameters
3D Observer Form [A(v),B(v),C(v),F(v),H(v)]	Implement a gain-scheduled state-space controller in an observer form depending on three scheduling parameters
3D Self-Conditioned [A(v),B(v),C(v),D(v)]	Implement a gain-scheduled state-space controller in a self-conditioned form

Gain Scheduled Lead-Lag	Implement a first-order lead-lag with gain-scheduled coefficients
Interpolate Matrix(x)	Return an interpolated matrix for given input x
Interpolate Matrix(x,y)	Return an interpolated matrix for given inputs x and y
Interpolate Matrix(x,y,z)	Return an interpolated matrix for given inputs x, y, and z
Self-Conditioned [A,B,C,D]	Implement a state-space controller in a self-conditioned form

Guidance Sublibrary

Calculate Range	Calculate the range between two crafts given their respective positions

Mass Properties Library

Estimate Center of Gravity	Calculate the center of gravity location
Estimate Inertia Tensor	Calculate the inertia tensor
Moments About CG Due to Forces	Compute moments about center of gravity due to forces that are applied at point CP, not the center of gravity
Symmetric Inertia Tensor	Create an inertia tensor from moments and products of inertia

Propulsion Library

Turbofan Engine System	Implement a first-order representation of a turbofan engine with controller

Utilities Library

The Utilities library contains the following sublibraries:

Axes Transformations Sublibrary

Direction Cosine Matrix to Euler Angles	Convert direction cosine matrix to Euler angles
Direction Cosine Matrix to Quaternions	Convert direction cosine matrix to quaternion vector
Euler Angles to Direction Cosine Matrix	Convert Euler angles to direction cosine matrix
Euler Angles to Quaternions	Convert Euler angles to quaternion vector
Quaternions to Direction Cosine Matrix	Convert quaternion vector to direction cosine matrix
Quaternions to Euler Angles	Convert quaternion vector to Euler angles

Math Operations Sublibrary

3x3 Cross Product	Calculate the cross product of two 3-by-1 vectors
Adjoint of 3x3 Matrix	Compute the adjoint matrix for the input matrix
Create 3x3 Matrix	Create a 3-by-3 matrix from nine input values
Determinant of 3x3 Matrix	Compute the determinant for the input matrix
Invert 3x3 Matrix	Compute the inverse of 3-by-3 matrix using determinant formula
SinCos	Compute the sine and cosine of input angle

Unit Conversions Sublibrary

Acceleration Conversion	Convert from acceleration units to desired acceleration units
Angle Conversion	Convert from angle units to desired angle units

Angular Acceleration Conversion	Convert from angular acceleration units to desired angular acceleration units
Angular Velocity Conversion	Convert from angular velocity units to desired angular velocity units
Density Conversion	Convert from density units to desired density units
Force Conversion	Convert from force units to desired force units
Length Conversion	Convert from length units to desired length units
Mass Conversion	Convert from mass units to desired mass units
Pressure Conversion	Convert from pressure units to desired pressure units
Temperature Conversion	Convert from temperature units to desired temperature units
Velocity Conversion	Convert from velocity units to desired velocity units

Blocks — Alphabetical List

This section contains block reference pages listed alphabetically.

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

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

Library   GNC/Controls

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

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

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

Dialog Box

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

StateSpaceABCD-1D (mask) (link)

Implement a state-space controller [A,B,C,D] where A, B, C, and D depend on one scheduling parameter, v.

Parameters

A-matrix(v):

A1

B-matrix(v):

B1

C-matrix(v):

C1

D-matrix(v):

D1

Scheduling variable breakpoints:

v_vec

Initial state, x_initial:

0

OK   Cancel   Help   Apply

A-matrix(v)

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

B-matrix(v)

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

C-matrix(v)

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

D-matrix(v)

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

Scheduling variable breakpoints

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

Initial state, x_initial

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

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

Inputs and Outputs	The first input is the measurements.
	The second input is the scheduling variable conforming to the dimensions of the state-space matrices.
	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.
Examples	See H-Infinity Controller (1 Dimensional Scheduling) in the aeroblk_lib_HL20 demo library 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)]

Purpose

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

Library

GNC/Controls

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

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

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

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.

4-15

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

Dialog Box

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

Blend-1D (mask) (link)

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

Parameters

A-matrix(v):

> A1

B-matrix(v):

> B1

C-matrix(v):

> C1

D-matrix(v):

> D1

Scheduling variable breakpoints:

> [1 1.5 2]

Initial state, x_initial:

> 0

Poles of A(v)-H(v)*C(v) = [w1 ... wn]:

> [-5 -2]

| OK | Cancel | Help | Apply |
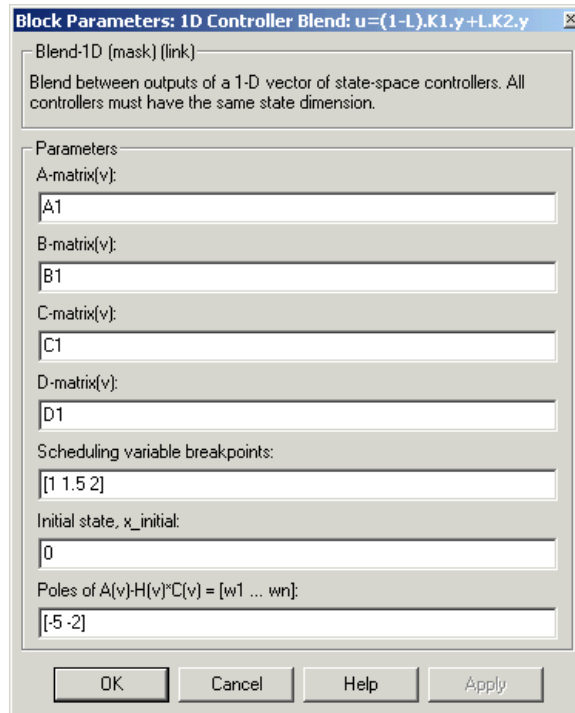
A-matrix(v)

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

The first input is the measurements.

The second input is the scheduling variable conforming to the dimensions of the state-space matrices.

The output is the actuator demands.

Assumptions and Limitations

This block requires the Control System Toolbox.

References

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

See Also

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

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

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

2D Controller Blend

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

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

Library   GNC/Controls

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 blocks is to implement a controller designed using H-infinity loop-shaping, one of the design methods supported by the μ-Analysis and Synthesis Toolbox.

Dialog Box

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

The first input is the set-point error.

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

The second input is the scheduling variable.

The third input is 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.

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

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

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

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

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

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

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

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

Purpose   Implement a gain-scheduled state-space controller in a self-conditioned form

Library   GNC/Controls

Description   The 1D Self-Conditioned [A(v),B(v),C(v),D(v)] block implements a gain-scheduled state-space controller as defined by the equations
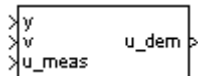
$$\dot{x} = A(v)x + B(v)y$$
$$u = C(v)x + D(v)y$$

in the self-conditioned form

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

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

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

Dialog Box

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

The first input is the measurements.

The second input is the scheduling variable conforming to the dimensions of the state-space matrices.

The third 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.

This block requires the Control System Toolbox.

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

References   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)]

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

Library   GNC/Controls

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.

Dialog Box

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

StateSpaceABCD-2D (mask) (link)

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

Parameters

A-matrix(v1,v2):

A

B-matrix(v1,v2):

B

C-matrix(v1,v2):

C

D-matrix(v1,v2):

D

First scheduling variable (v1) breakpoints:

v1_vec

Second scheduling variable (v2) breakpoints:

v2_vec

Initial state, x_initial:

0

OK   Cancel   Help   Apply

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

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 v1 and v2. Hence, for example, if the B-matrix corresponding to the first entry of v1 and first entry of v2 is the identity matrix, then B(:,:,1,1) = [1 0;0 1];.

C-matrix(v1,v2)

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

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 v1 and v2. Hence, for example, if the D-matrix corresponding to the first entry of v1 and first entry of v2 is the identity matrix, then D(:,:,1,1) = [1 0;0 1];.

First scheduling variable (v1) breakpoints

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

Second scheduling variable (v2) breakpoints

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

Initial state, x_initial

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

Inputs and Outputs

The first input is the measurements.

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

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.

Examples

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

See Also

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

2D Controller Blend

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

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

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

2D Controller Blend

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

Library   GNC/Controls

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

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

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

Dialog Box

A-matrix(v1,v2)

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

B-matrix(v1,v2)

> B-matrix of the state-space implementation.

C-matrix(v1,v2)

> C-matrix of the state-space implementation.

2D Controller Blend

D-matrix(v1,v2)

D-matrix of the state-space implementation.

First scheduling variable (v1) breakpoints

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

Second scheduling variable (v2) breakpoints

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

Initial state, x_initial

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

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

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

Inputs and Outputs

The first input is the measurements.

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

The output is the actuator demands.

Assumptions and Limitations

This block requires the Control System Toolbox.

References

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

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

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

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

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

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

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

Library　　　　GNC/Controls

Description　　The 2D Observer Form [A(v),B(v),C(v),F(v),H(v)] block implements a gain-scheduled state-space controller defined in the following observer form:
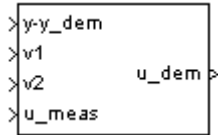
$$\dot{x} = (A(v) + H(v)C(v))x + B(v)u_{meas} + H(v)(y - y_{dem})$$
$$u_{dem} = F(v)x$$

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

Dialog Box

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

┌ StateSpaceABCFH-2D (mask) (link) ─────────────────────────
Implement a state-space controller [A,B,C,F,H] in observer form where A,
B, C, F, and H depend on two scheduling parameters.

┌ Parameters ──
A-matrix(v1,v2):

    A

B-matrix(v1,v2):

    B

C-matrix(v1,v2):

    C

F-matrix(v1,v2):

    F

H-matrix(v1,v2):

    H

First scheduling variable (v1) breakpoints:

    v1_vec

Second scheduling variable (v2) breakpoints:

    v2_vec

Initial state, x_initial:

    0

   OK   Cancel   Help   Apply

A-matrix(v1,v2)

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

B-matrix(v1,v2)

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

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

C-matrix(v1,v2)

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

F-matrix(v1,v2)

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

H-matrix(v1,v2)

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

First scheduling variable (v1) breakpoints

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

Second scheduling variable (v2) breakpoints

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

The first input is the set-point error.

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

The fourth 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.

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

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

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

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

2D Controller Blend

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

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

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

Purpose   Implement a gain-scheduled state-space controller in a self-conditioned form

Library   GNC/Controls

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

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

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

Dialog Box

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

StateSpaceSelfCond-2D (mask) (link)

Implement a state-space controller [A(v1,v2),B(v1,v2),C(v1,v2),D(v1,v2)] in a self-conditioned form. If u_meas = u_dem, then the implemented controller is [A,B,C,D]. If u_meas is limited, e.g., rate limiting, then the poles of the controller become those defined in the mask dialog box. Uses call to Control Systems Toolbox function place.m when initializing. A, B, C, and D should be 4-D matrices, the last two dimensions corresponding to the scheduling parameters, and the first two corresponding to the matrix for a given set of scheduling parameter values.

Parameters

A-matrix(v1,v2):

| A |

B-matrix(v1,v2):

| B |

C-matrix(v1,v2):

| C |

D-matrix(v1,v2):

| D |

First scheduling variable (v1) breakpoints:

| v1_vec |

Second scheduling variable (v2) breakpoints:

| v2_vec |

Initial state, x_initial:

| 0 |

Poles of A(v)-H(v)*C(v) = [w1 ... wn]:

| [-5 -2] |

   OK   Cancel   Help   Apply

A-matrix(v1,v2)

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

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

B-matrix(v1,v2)

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

C-matrix(v1,v2)

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

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 v1 and v2. Hence, for example, if the D-matrix corresponding to the first entry of v1 and first entry of v2 is the identity matrix, then D(:,:,1,1) = [1 0;0 1];.

First scheduling variable (v1) breakpoints

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

Second scheduling variable (v2) breakpoints

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

Initial state, x_initial

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

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

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

The first input is the measurements.

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

The fourth 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.

This block requires the Control System Toolbox.

References

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

See Also

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

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

2D Controller Blend

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

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

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

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

Library   GNC/Controls

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.

Dialog Box

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;0 1 0; 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 v1, v2, and v3. Hence, for example, if the B-matrix

corresponding to the first entry of v1, the first entry of v2, and the first entry of v3 is the identity matrix, then B(:,:,1,1,1) = [1 0;0 1];.

C-matrix(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 v1, v2, and v3. Hence, for example, if the C-matrix corresponding to the first entry of v1, the first entry of v2, and the first entry of v3 is the identity matrix, then C(:,:,1,1,1) = [1 0;0 1];.

D-matrix(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 v1, v2, and v3. Hence, for example, if the D-matrix corresponding to the first entry of v1, the first entry of v2, and the first entry of v3 is the identity matrix, then D(:,:,1,1,1) = [1 0;0 1];.

First scheduling variable (v1) breakpoints

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

Second scheduling variable (v2) breakpoints

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

Third scheduling variable (v3) breakpoints

Vector of the breakpoints for the third scheduling variable. The length of v3 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

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 output is the actuator demands.

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

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

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

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

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

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

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

Library   GNC/Controls

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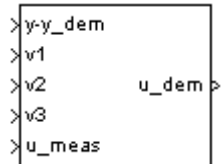
$$\dot{x} = (A(v) + H(v)C(v))x + B(v)u_{meas} + H(v)(y - y_{dem})$$

$$u_{dem} = F(v)x$$

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

Dialog Box

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

StateSpaceABCFH-3D (mask) (link)

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

Parameters

A-matrix(v1,v2,v3):

A

B-matrix(v1,v2,v3):

B

C-matrix(v1,v2,v3):

C

F-matrix(v1,v2,v3):

F

H-matrix(v1,v2,v3):

H

First scheduling variable (v1) breakpoints:

v1_vec

Second scheduling variable (v2) breakpoints:

v2_vec

Third scheduling variable (v3) breakpoints:

v3_vec

Initial state, x_initial:

0

OK   Cancel   Help   Apply

A-matrix(v1,v2,v3)

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

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 v1, v2, and v3. Hence, for example, if the B-matrix corresponding to the first entry of v1, the first entry of v2, and the first entry of v3 is the identity matrix, then B(:,:,1,1,1) = [1 0;0 1];.

C-matrix(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 v1, v2, and v3. Hence, for example, if the C-matrix corresponding to the first entry of v1, the first entry of v2, and the first entry of v3 is the identity matrix, then C(:,:,1,1,1) = [1 0;0 1];.

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 v1, v2, and v3. Hence, for example, if the F-matrix corresponding to the first entry of v1, the first entry of v2, and the first entry of v3 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 v1, v2, and v3. Hence, for example, if the H-matrix corresponding to the first entry of v1, the first entry of v2, and the first entry of v3 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 v1 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 v2 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 v3 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.

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

Inputs and Outputs

The first input is the set-point error.

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

The fifth input is 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.

References

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

See Also

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

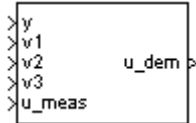2D Observer Form [A(v),B(v),C(v),F(v),H(v)]

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

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

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

Purpose   Implement a gain-scheduled state-space controller in a self-conditioned form

Library   GNC/Controls

Description

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

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

in the self-conditioned form

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

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

Dialog Box

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

StateSpaceSelfCond-3D (mask) (link)

Implement a state-space controller
[A(v1,v2,v3),B(v1,v2,v3),C(v1,v2,v3),D(v1,v2,v3)] in a self-conditioned
form. If u_meas = u_dem, then the implemented controller is [A,B,C,D]. If
u_meas is limited, e.g., rate limiting, then the poles of the controller
become those defined in the mask dialog box. Uses call to Control
Systems Toolbox function place.m when initializing. A, B, C, and D should
be 5-D matrices, the last three dimensions corresponding to the
scheduling parameters, and the first two corresponding to the matrix for a
given set of scheduling parameter values.

Parameters

A-matrix(v1,v2,v3):

    A

B-matrix(v1,v2,v3):

    B

C-matrix(v1,v2,v3):

    C

D-matrix(v1,v2,v3):

    D

First scheduling variable (v1) breakpoints:

    v1_vec

Second scheduling variable (v2) breakpoints:

    v2_vec

Third scheduling variable (v3) breakpoints:

    v3_vec

Initial state, x_initial:

    0

Poles of A(v)-H(v)*C(v) = [w1 ... wn]:

    [-5 -2]

    OK   Cancel   Help   Apply

A-matrix(v1,v2,v3)

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

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 v1, v2, and v3. Hence, for example, if the B-matrix corresponding to the first entry of v1, the first entry of v2, and the first entry of v3 is the identity matrix, then B(:,:,1,1,1) = [1 0;0 1];.

C-matrix(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 v1, v2, and v3. Hence, for example, if the C-matrix corresponding to the first entry of v1, the first entry of v2, and the first entry of v3 is the identity matrix, then C(:,:,1,1,1) = [1 0;0 1];.

D-matrix(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 v1, v2, and v3. Hence, for example, if the D-matrix corresponding to the first entry of v1, the first entry of v2, and the first entry of v3 is the identity matrix, then D(:,:,1,1,1) = [1 0;0 1];.

First scheduling variable (v1) breakpoints

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

Second scheduling variable (v2) breakpoints

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

Third scheduling variable (v3) breakpoints

Vector of the breakpoints for the third scheduling variable. The length of v3 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

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.

This block requires the Control System Toolbox.

References

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

See Also

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

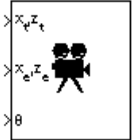2D Self-Conditioned [A(v),B(v),C(v),D(v)]

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

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

3DoF Animation

Purpose   Create a 3-D Handle Graphics animation of a three-degrees-of-freedom object

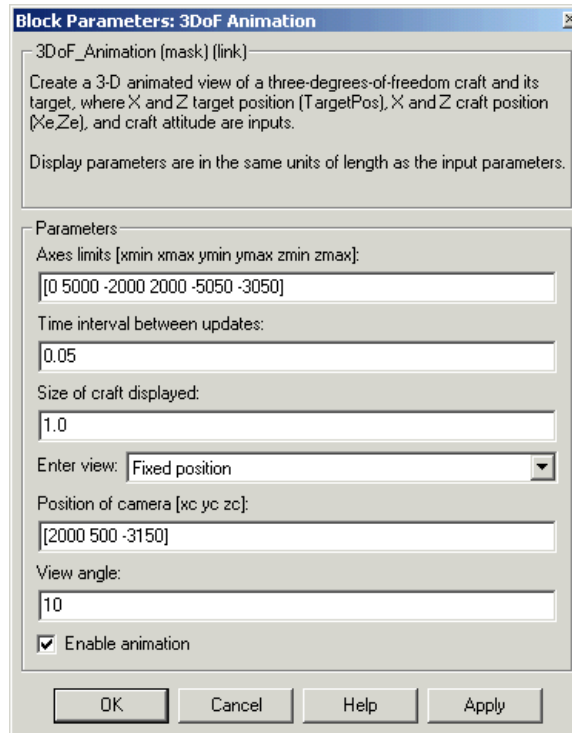**Library**   Animation

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
Handle Graphics.

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

Dialog Box

Axes limits [xmin xmax ymin ymax zmin zmax]

Specifies the three-dimensional space to be viewed.

Time interval between updates

Specifies the time interval at which the animation is redrawn.

Size of craft displayed

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

Enter view

Selects preset Handle 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 Handle Graphics parameter **CameraPosition** for the figure axes. Used in all cases except for the Cockpit view.

View angle

Specifies the Handle 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   The first input is a vector containing the altitude and the downrange position of the target in Earth coordinates.

The second input is a vector containing the altitude and the downrange position of the craft in Earth coordinates.

The third input is the attitude of the craft.

3DoF Animation

Examples   See the `aero_guidance` demo for an example of this block.
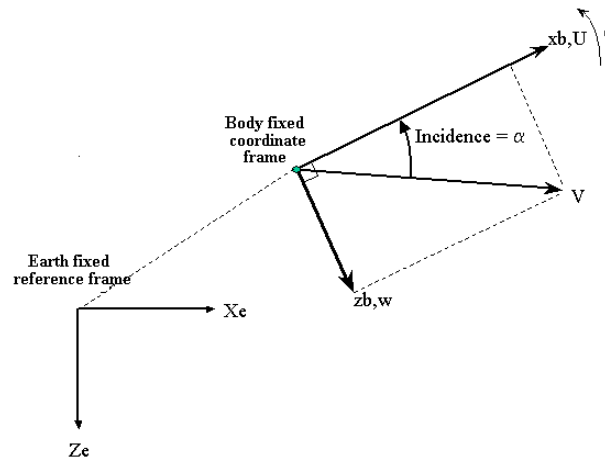
See Also   6DoF Animation
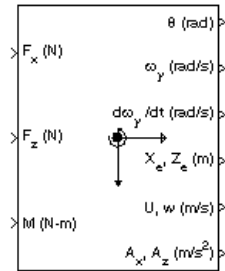
Purpose   Implement three-degrees-of-freedom equations of motion

Library   Equations of Motion/3DoF

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

The equations of motion are

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

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

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

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

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

4-55

3DoF (Body Axes)

Dialog Box

Block Parameters: 3DoF (Body Axes)

3DoF EoM (mask) (link)

Integrate the three-degrees-of-freedom equations of motion to determine body position, velocity, attitude, and related values.

Parameters

Units: Metric (MKS)

Mass type: Fixed

Initial velocity:

100

Initial body attitude:

0

Initial incidence:

0

Initial body rotation rate:

0

Initial position (x z):

[0 0]

Initial mass:

1.0

Inertia:

1.0

Gravity source: External

OK   Cancel   Help   Apply

Units

Specifies the input and output 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:

Mass	Description
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).

3DoF (Body Axes)

Initial body rotation rate

A scalar value for the initial body rotation rate, (q_0).

Initial position (x,z)

A two-element vector containing the initial location of the body in the Earth-fixed reference frame.

Initial Mass

A scalar value for the mass of the body.

Inertia

A scalar value for the inertia of the body.

Gravity Source

Specify source of gravity:

External	Variable gravity input to block
Internal	Constant gravity specified in mask

Acceleration due to gravity

A scalar value for the acceleration due to gravity used if internal gravity source is selected. If gravity is to be neglected in the simulation, this value can be set to 0.

Inputs and Outputs

The first input to the block is the force acting along the body x-axis, (F_x).

The second input to the block is the force acting along the body z-axis, (F_z).

The third input to the block is the applied pitch moment, (M).

The fourth optional input to the block is gravity in the selected units.

The first output from the block is the pitch attitude, in radians (θ).

The second output is the pitch angular rate, in radians per second (q).

The third output is the pitch angular acceleration, in radians per second squared $(\dot{q})$.

The fourth output is a two-element vector containing the location of the body, in the Earth-fixed reference frame, (Xe, Ze).

The fifth output is a two-element vector containing the velocity of the body resolved into the body-fixed coordinate frame, (*u,w*).

The sixth output is a two-element vector containing the acceleration of the body resolved into the body-fixed coordinate frame, (*Ax,Az*).

Examples   See the `aero_guidance` demo for an example of this block.

See Also   Custom Variable Mass 3DoF (Body Axes)

Incidence & Airspeed

Simple Variable Mass 3DoF (Body Axes)

3x3 Cross Product

Purpose   Calculate the cross product of two 3-by-1 vectors

Library   Utilities/Math Operations
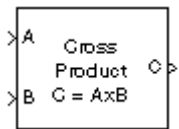
Description   The 3x3 Cross Product block computes cross (or vector) product of two vectors, A and B, by generating a third vector, C, in a direction normal to the plane containing A and B, and with magnitude equal to the product of the lengths of A and B multiplied by the sine of the angle between them. The direction of C is that in which a right-handed screw would move in turning from A to B.
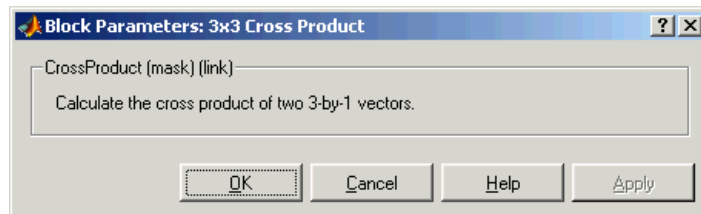
$$A = a_1\mathbf{i} + a_2\mathbf{j} + a_3\mathbf{k}$$
$$B = b_1\mathbf{i} + b_2\mathbf{j} + b_3\mathbf{k}$$

$$C = A \times B = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix}$$

$$= (a_2b_3 - a_3b_2)\mathbf{i} + (a_3b_1 - a_1b_3)\mathbf{j} + (a_1b_2 - a_2b_1)\mathbf{k}$$

Dialog Box

Inputs and Outputs   The inputs are two 3-by-1 vectors.
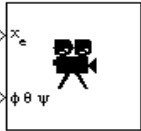
The output is a 3-by-1 vector.

Purpose   Create a 3-D Handle Graphics® animation of a six-degrees-of-freedom object
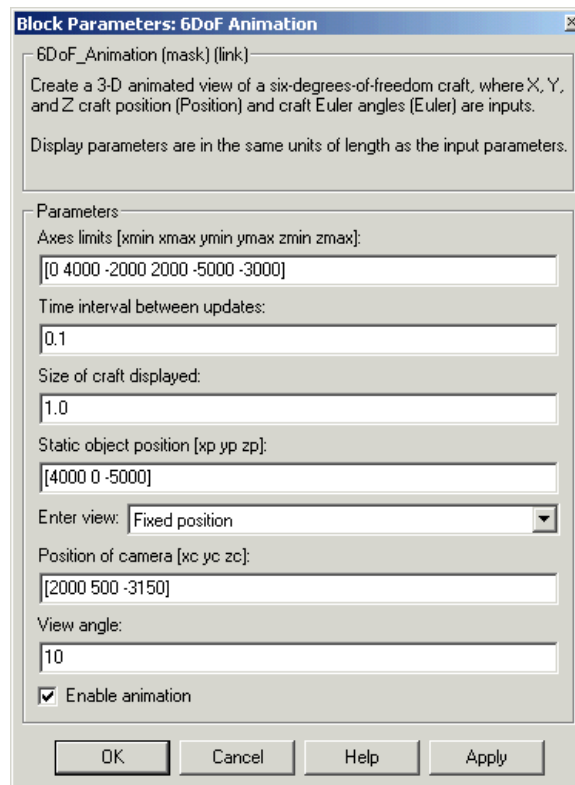
Library   Animation

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 Handle Graphics.

The 6DoF Animation block uses the input values and the dialog parameters to create and display the animation.

Dialog Box

6DoF Animation

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 cross-range position, and the downrange position of the target.

Enter view

Selects preset Handle 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 Handle Graphics parameter **CameraPosition** for the figure axes. Used in all cases except for the Cockpit view.

View angle

Specifies the Handle 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   The first input is a vector containing the altitude, the cross-range position, and the downrange position of the craft in Earth coordinates.

The second input is a vector containing the Euler angles of the craft.
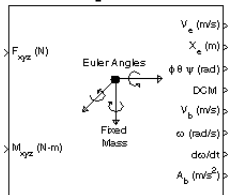
See Also   3DoF Animation

Purpose   Implement an Euler angle representation of six-degrees-of-freedom equations of motion
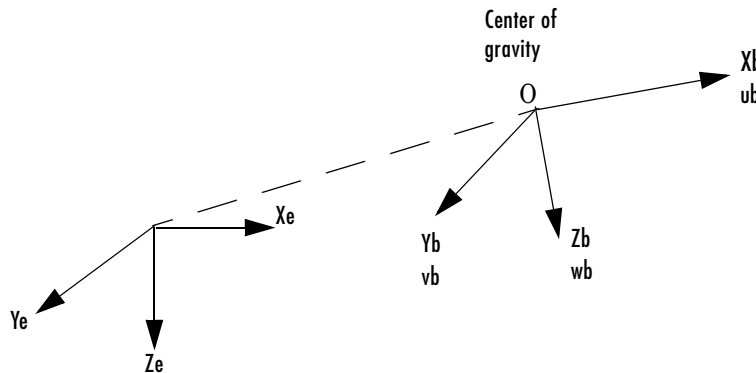
Library   Equations of Motion/6DoF

Description   The 6DoF (Euler Angles) block considers the rotation of a body-fixed coordinate frame (X_b, Y_b, Z_b) about an Earth-fixed 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 Earth-fixed reference frame is considered inertial, a simplification that allows the forces due to the Earth's motion relative to a star-fixed reference system to be neglected.

Earth-fixed 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.

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

6DoF (Euler Angles)

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

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

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

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix}$$
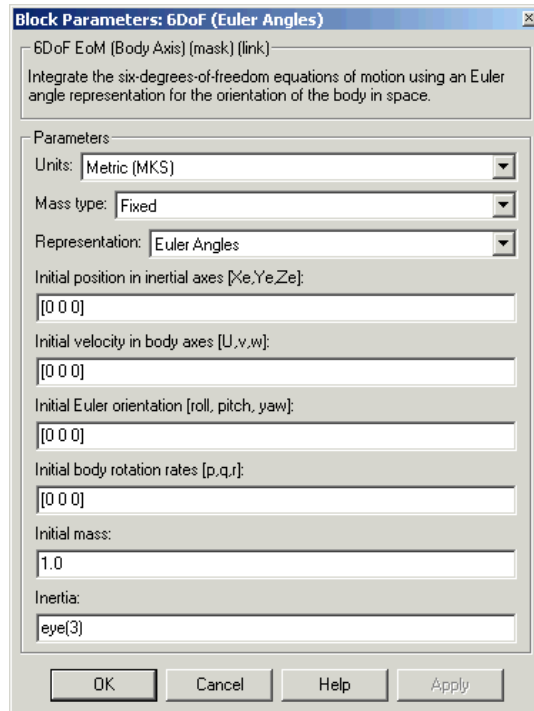
The relationship between the body-fixed angular velocity vector, $[p\ q\ r]^T$, and the rate of change of the Euler angles, $[\ \dot{\phi}\ \dot{\theta}\ \dot{\psi}\ ]^T$, can be determined by resolving the Euler rates into the body-fixed coordinate frame.

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

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

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

Dialog Box

Block Parameters: 6DoF (Euler Angles)

─ 6DoF EoM (Body Axis) (mask) (link) ─

Integrate the six-degrees-of-freedom equations of motion using an Euler angle representation for the orientation of the body in space.

─ Parameters ─

Units: Metric (MKS)

Mass type: Fixed

Representation: Euler Angles

Initial position in inertial axes [Xe,Ye,Ze]:

[0 0 0]

Initial velocity in body axes [U,v,w]:

[0 0 0]

Initial Euler orientation [roll, pitch, yaw]:

[0 0 0]

Initial body rotation rates [p,q,r]:

[0 0 0]

Initial mass:

1.0

Inertia:

eye(3)

OK   Cancel   Help   Apply

Units

Specifies the input and output units:

	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

6DoF (Euler Angles)

Mass Type

Select the type of mass to use:

Mass	Description
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:

Mass	Description
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 Earth-fixed reference frame.

Initial velocity in body axes

The three-element vector for the initial velocity in the body-fixed coordinate frame.

Initial Euler rotation

The three-element vector for the initial Euler rotation angles [roll, pitch, yaw], in radians.

Initial body rotation rates

The three-element vector for the initial body-fixed angular rates, in radians per second.

Initial Mass

The mass of the rigid body.

Inertia

The 3-by-3 inertia tensor matrix I.

Inputs and Outputs

The first input to the block is a vector containing the three applied forces.

The second input is a vector containing the three applied moments.

The first output is a three-element vector containing the velocity in the Earth-fixed reference frame.

The second output is a three-element vector containing the position in the Earth-fixed reference frame.

The third output is a three-element vector containing the Euler rotation angles [roll, pitch, yaw], in radians.

The fourth output is a 3-by-3 matrix for the coordinate transformation from Earth-fixed axes to body-fixed axes.

The fifth output is a three-element vector containing the velocity in the body-fixed frame.

The sixth output is a three-element vector containing the angular rates in body-fixed axes, in radians per second.

The seventh output is a three-element vector containing the angular accelerations in body-fixed axes, in radians per second.

The eighth output is a three-element vector containing the accelerations in body-fixed axes.

Assumptions and Limitations

The block assumes that the applied forces are acting at the center of gravity of the body, and that the mass and inertia are constant.

Examples

See the aeroblk_six_dof demo and Airframe in the aeroblk_HL20 demo for examples of this block.

References

Mangiacasale, L., "Flight Mechanics of a u-Airplane with a MATLAB Simulink Helper," Edizioni Libreria CLUP, 1998.

6DoF (Euler Angles)

See Also

6DoF (Quaternion)

Custom Variable Mass 6DoF (Euler Angles)

Custom Variable Mass 6DoF (Quaternion)

Simple Variable Mass 6DoF (Euler Angles)

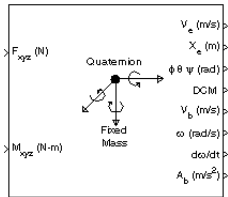Simple Variable Mass 6DoF (Quaternion)

Purpose   Implement a quaternion 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 (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.

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

$$\varepsilon = 1 - (q_0^2 + q_1^2 + q_3^2 + q_4^2)$$

6DoF (Quaternion)

Dialog Box

Block Parameters: 6DoF (Quaternion)

6DoF EoM (Body Axis) (mask) (link)

Integrate the six-degrees-of-freedom equations of motion using an Euler angle representation for the orientation of the body in space.

Parameters

Units: Metric (MKS)

Mass type: Fixed

Representation: Quaternion

Initial position in inertial axes [Xe,Ye,Ze]:

[0 0 0]

Initial velocity in body axes [U,v,w]:

[0 0 0]

Initial Euler orientation [roll, pitch, yaw]:

[0 0 0]

Initial body rotation rates [p,q,r]:

[0 0 0]

Initial mass:

1.0

Inertia:

eye(3)

Gain for quaternion normalization:

1.0

OK   Cancel   Help   Apply

Units

Specifies the input and output 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:

Mass	**Description**
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:

Mass	**Description**
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.

6DoF (Quaternion)

Initial position in inertial axes

The three-element vector for the initial location of the body in the Earth-fixed reference frame.

Initial velocity in body axes

The three-element vector for the initial velocity in the body-fixed coordinate frame.

Initial Euler rotation

The three-element vector for the initial Euler rotation angles [roll, pitch, yaw], in radians.

Initial body rotation rates

The three-element vector for the initial body-fixed angular rates, in radians per second.

Initial Mass

The mass of the rigid body.

Inertia matrix

The 3-by-3 inertia tensor matrix I.

Gain for quaternion normalization

The gain to maintain the norm of the quaternion vector equal to 1.0.

Inputs and Outputs

The first input to the block is a vector containing the three applied forces.

The second input is a vector containing the three applied moments.

The first output is a three-element vector containing the velocity in the Earth-fixed reference frame.

The second output is a three-element vector containing the position in the Earth-fixed reference frame.

The third output is a three-element vector containing the Euler rotation angles [roll, pitch, yaw], in radians.

The fourth output is a 3-by-3 matrix for the coordinate transformation from Earth-fixed axes to body-fixed axes.

The fifth output is a three-element vector containing the velocity in the body-fixed frame.

The sixth output is a three-element vector containing the angular rates in body-fixed axes, in radians per second.

The seventh output is a three-element vector containing the angular accelerations in body-fixed axes, in radians per second.

The eighth output is a three-element vector containing the accelerations in body-fixed axes.

Assumptions and Limitations   The block assumes that the applied forces are acting at the center of gravity of the body, and that the mass and inertia are constant.

References   Mangiacasale, L., "Flight Mechanics of a u-Airplane with a MATLAB Simulink Helper," Edizioni Libreria CLUP, 1998.

See Also   6DoF (Euler Angles)

Custom Variable Mass 6DoF (Euler Angles)

Custom Variable Mass 6DoF (Quaternion)

Simple Variable Mass 6DoF (Euler Angles)

Simple Variable Mass 6DoF (Quaternion)

Acceleration Conversion

Purpose   Convert from acceleration units to desired acceleration units

Library   Utilities/Unit Conversions
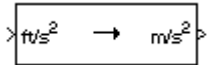
Description   The Acceleration Conversion block computes the conversion factor from

specified input acceleration units to specified output acceleration units and
applies the conversion factor to the input signal.

The Acceleration Conversion block icon displays the input and output units
selected from the **Initial units** and **Final units** lists.
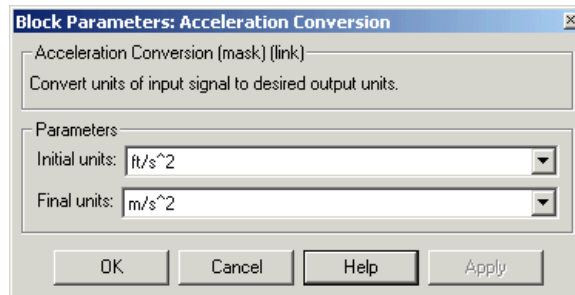
Dialog Box

Initial units

   Specifies the input units.

Final units

   Specifies the output units.

The following conversion units are available:

m/s^2	Meters per second squared
ft/s^2	Feet per second squared
km/s^2	Kilometers per second squared
in/s^2	Inches per second squared
km/h-s	Kilometers per hour per second
mph-s	Miles per hour per second

Acceleration Conversion

Inputs and Outputs

The input is acceleration in initial acceleration units.

The output is acceleration in final acceleration units.

See Also

Angle Conversion

Angular Acceleration Conversion

Angular Velocity Conversion

Density Conversion

Force Conversion

Length Conversion

Mass Conversion

Pressure Conversion

Temperature Conversion

Velocity Conversion

Adjoint of 3x3 Matrix

Purpose   Compute the adjoint matrix for the input matrix

Library   Utilities/Math Operations

Description   The Adjoint of 3x3 Matrix block computes the adjoint matrix for the input matrix.

adj(A)
(3x3)

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

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

Dialog Box

Block Parameters: Adjoint of 3x3 Matrix

─ Adjoint of 3x3 matrix (mask) (link) ─

Compute the adjoint matrix for the input matrix.

OK   Cancel   Help   Apply
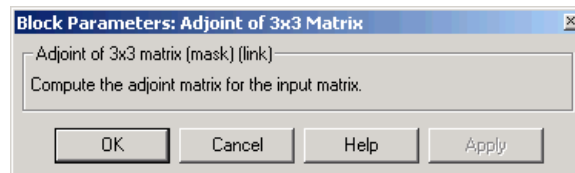
Inputs and Outputs

The input is a 3-by-3 matrix.

The output of the block is 3-by-3 adjoint matrix of input matrix.

See Also   Create 3x3 Matrix

Determinant of 3x3 Matrix

Invert 3x3 Matrix
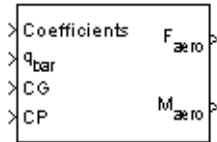
Aerodynamic Forces and Moments

Purpose   Compute the aerodynamic forces and moments using the aerodynamic coefficients, dynamic pressure, center of gravity, and center of pressure.
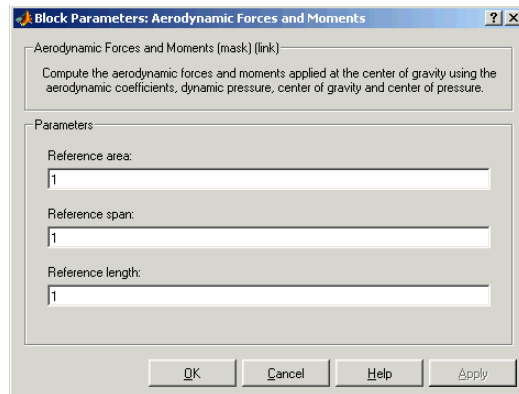
Library   Aerodynamics

Description The Aerodynamic Forces and Moments block computes the aerodynamic forces and moments about the center of gravity.

Dialog Box

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 is aerodynamic coefficients (in body axes) for forces and moments.

The second input is the dynamic pressure.

The third input is the center of gravity.

The fourth input is the center of pressure.

The first output of the block is aerodynamic forces at the center of gravity in x-axes, y-axes and z-axes.

The second output of the block is aerodynamic moments at the center of gravity in x-axes, y-axes and z-axes.

Examples See Airframe in the aeroblk_HL20 demo for an example of this block.

See Also Dynamic Pressure

Estimate Center of Gravity

Moments About CG Due to Forces

Angle Conversion

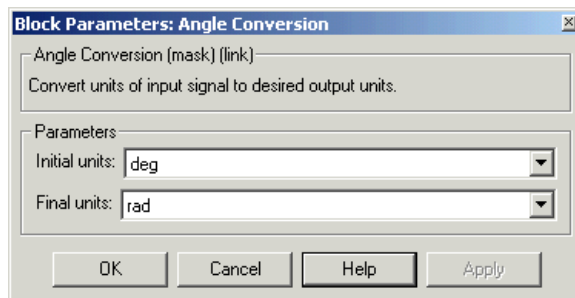**Purpose**   Convert from angle units to desired angle units

Library   Utilities/Unit Conversions

Description   The Angle Conversion block computes the conversion factor from specified input angle units to specified output angle units and applies the conversion factor to the input signal.
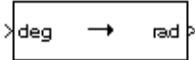
The Angle Conversion block icon displays the input and output units selected from the **Initial units** and the **Final units** lists.

Dialog Box

Block Parameters: Angle Conversion

Angle Conversion (mask) (link)

Convert units of input signal to desired output units.

Parameters

Initial units: deg

Final units: rad

| OK | Cancel | Help | Apply |

Initial units
   Specifies the input units.

Final units
   Specifies the output units.

The following conversion units are available:

deg	Degrees
rad	Radians
rev	Revolutions

Inputs and Outputs   The input is angle in initial angle units.

The output is angle in final angle units.

See Also   Acceleration Conversion

Angular Acceleration Conversion

Angular Velocity Conversion

Density Conversion

Force Conversion

Length Conversion

Mass Conversion

Pressure Conversion

Temperature Conversion

Velocity Conversion

Angular Acceleration Conversion

**Purpose**   Convert from angular acceleration units to desired angular acceleration units

Library   Utilities/Unit Conversions

Description   The Angular Acceleration Conversion block computes the conversion factor
from specified input angular acceleration units to specified output angular
acceleration units and applies the conversion factor to the input signal.
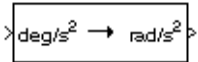
$$\text{deg/s}^2 \rightarrow \text{rad/s}^2$$

The Angular Acceleration Conversion block icon displays the input and output
units selected from the **Initial units** and the **Final units** lists.

Dialog Box
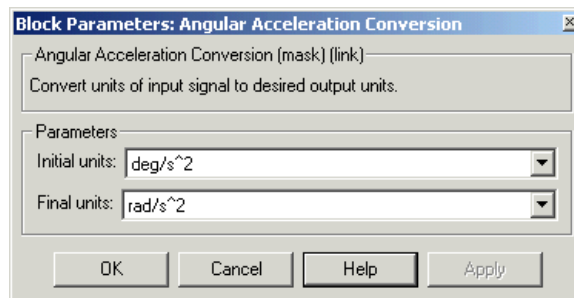
Initial units
Specifies the input units.

Final units
Specifies the output units.

The following conversion units are available:

deg/s^2	Degrees per second squared
rad/s^2	Radians per second squared
rpm/s	Revolutions per minute per second

Inputs and Outputs   The input is angular acceleration in initial angular acceleration units.

The output is angular acceleration in final angular acceleration units.

See Also   Acceleration Conversion

Angle Conversion

Angular Velocity Conversion

Density Conversion

Force Conversion

Length Conversion

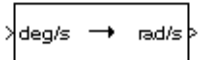Mass Conversion

Pressure Conversion

Temperature Conversion

Velocity Conversion

Angular Velocity Conversion

Purpose   Convert from angular velocity units to desired angular velocity units
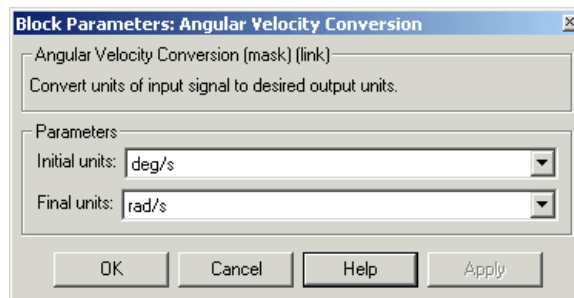
Library   Utilities/Unit Conversions

Description

| deg/s → rad/s |

The Angular Velocity Conversion block computes the conversion factor from specified input angular velocity units to specified output angular velocity units and applies the conversion factor to the input signal.

The Angular Velocity Conversion block icon displays the input and output units selected from the **Initial units** and the **Final units** lists.

Dialog Box

```
Block Parameters: Angular Velocity Conversion            ×
─Angular Velocity Conversion (mask) (link)────────────────
 Convert units of input signal to desired output units.

─Parameters──────────────────────────────────────────────
 Initial units:  deg/s                                  ▼

 Final units:  rad/s                                    ▼


        OK      │   Cancel  │    Help    │    Apply
```

Initial units
> Specifies the input units.

Final units
> Specifies the output units.

The following conversion units are available:

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

Inputs and Outputs

The input is angular velocity in initial angular velocity units.

The output is angular velocity in final angular velocity units.

See Also   Acceleration Conversion

Angle Conversion

Angular Acceleration Conversion

Density Conversion

Force Conversion

Length Conversion

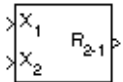Mass Conversion

Pressure Conversion

Temperature Conversion

Velocity Conversion

Calculate Range

Purpose
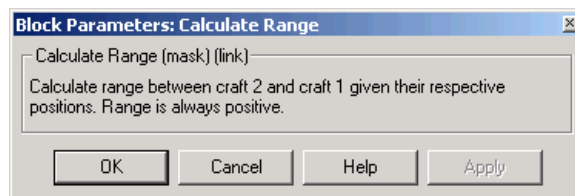Calculate the range between two crafts given their respective positions.

Library
GNC/Guidance

Description
The Calculate Range block computes the range between two crafts. The equation used for the range calculation is

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

Dialog Box

Inputs and Outputs
The first input is the (x, y and z) position of craft 1.

The second input is the (x, y and z) position of craft 2.

The output is the range from craft 2 and craft 1.

Limitation
The calculated range is give the magnitude of the distance but not the direction therefore it is always positive.

Craft positions are real values.

Purpose   Implement the 1976 COESA lower atmosphere

**Library**   Environment/Atmosphere
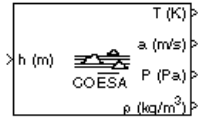
Description

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 32000 meters (approximately 104987 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.

Dialog Box
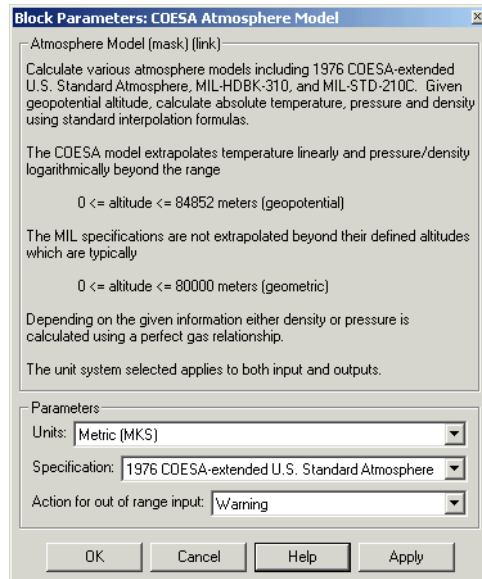
COESA Atmosphere Model

Units

Specifies the input and output units:

	Height	Temperature	Speed of Sound	Air Pressure	Air Density
Metric (MKS)	Meters	Degrees Kelvin	Meters per second	Pascal	Kilograms per cubic meter
English (Velocity in ft/s)	Feet	Degrees Rankine	Feet per second	Pound-force per square inch	Slug per cubic foot
English (Velocity in kts)	Feet	Degrees Rankine	Knots	Pound-force per square inch	Slug per cubic foot

Specification

Specify the atmosphere model type from one of the following atmosphere models. The default is 1976 COESA-extended U.S. Standard Atmosphere.

MIL-HDBK-310
This selection is linked to the Non-Standard Day 310 block. See the block reference for more information.

MIL-STD-210C
This selection is linked to the Non-Standard Day 210C block. See the block reference for more information.

Action for out of range input

Specify if out of range input invokes a warning, error, or no action.

Inputs and Outputs

The input is geopotential height.

The four outputs are temperature, speed of sound, air pressure, and air density.

Assumptions and Limitations

Below the geopotential altitude of 0 m (0 feet) and above the geopotential altitude of 84852 m (approximately 278386 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 Airframe in the `aeroblk_HL20` demo for examples of this block.

References   U.S. Standard Atmosphere, 1976, U.S. Government Printing Office, Washington, D.C.

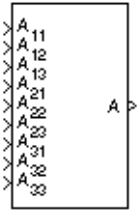**See Also**   ISA Atmosphere Model

Non-Standard Day 210C

Non-Standard Day 310

Create 3x3 Matrix

Purpose   Create a 3-by-3 matrix from nine input values.

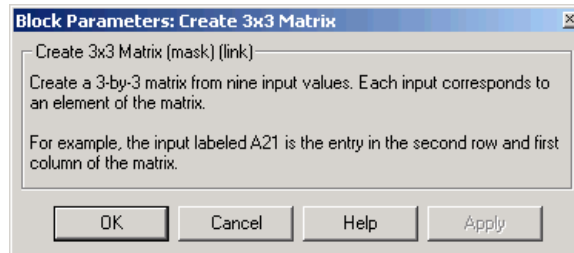**Library**   Utilities/Math Operations

Description   The Create 3x3 Matrix block creates a 3-by-3 matrix from nine input values where each input corresponds to an element of the matrix.

The output matrix has the form of

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

Dialog Box

Block Parameters: Create 3x3 Matrix

Create 3x3 Matrix (mask) (link)

Create a 3-by-3 matrix from nine input values. Each input corresponds to an element of the matrix.

For example, the input labeled A21 is the entry in the second row and first column of the matrix.

| OK | Cancel | Help | Apply |

Inputs and Outputs

The first input is the entry of the first row and first column of the matrix.

The second input is the entry of the first row and second column of the matrix.

The third input is the entry of the first row and third column of the matrix.

The fourth input is the entry of the second row and first column of the matrix.

The fifth input is the entry of the second row and second column of the matrix.

The sixth input is the entry of the second row and third column of the matrix.

The seventh input is the entry of the third row and first column of the matrix.

The eighth input is the entry of the third row and second column of the matrix.

The ninth input is the entry of the third row and third column of the matrix.

The output of the block is a 3-by-3 matrix.

See Also   Adjoint of 3x3 Matrix

Determinant of 3x3 Matrix

Invert 3x3 Matrix

Symmetric Inertia Tensor

Custom Variable Mass 3DoF (Body Axes)

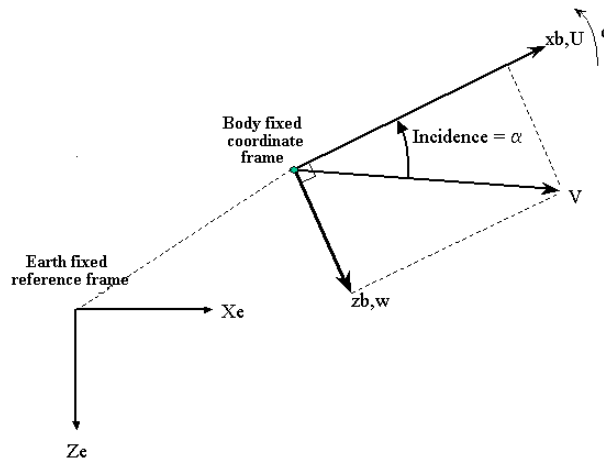**Purpose**   Implement three-degrees-of-freedom equations of motion

Library   Equations of Motion/3DoF

Description

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

The equations of motion are

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

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

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

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

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

Dialog Box

Custom Variable Mass 3DoF (Body Axes)

Units

Specifies the input and output 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:

Mass	**Description**
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 Earth-fixed reference frame.

Gravity Source

Specify source of gravity:

External	Variable gravity input to block
Internal	Constant gravity specified in mask

Acceleration due to gravity

A scalar value for the acceleration due to gravity used if internal gravity source is selected. If gravity is to be neglected in the simulation, this value can be set to 0.

Inputs and Outputs

The first input to the block is the force acting along the body *x-axis*, (F_x).

The second input to the block is the force acting along the body *z-axis*, (F_z).

The third input to the block is the applied pitch moment, (M).

The fourth input to the block is the rate of change of mass, $(\dot{m})$.

The fifth input to the block is the mass, (m).

The sixth input to the block is the rate of change of inertia tensor matrix, $(\dot{I}_{yy})$.

The seventh input to the block is the inertia tensor matrix, (I_{yy}).

The eighth optional input to the block is gravity in the selected units.

The first output from the block is the pitch attitude, in radians (θ).

The second output is the pitch angular rate, in radians per second (q).

The third output is the pitch angular acceleration, in radians per second squared $(\dot{q})$.

The fourth output is a two-element vector containing the location of the body, in the Earth-fixed reference frame, (Xe, Ze).

Custom Variable Mass 3DoF (Body Axes)

The fifth output is a two-element vector containing the velocity of the body resolved into the body-fixed coordinate frame, (u,w).

The sixth output is a two-element vector containing the acceleration of the body resolved into the body-fixed coordinate frame, (Ax,Az).

See Also

3DoF (Body Axes)

Incidence & Airspeed

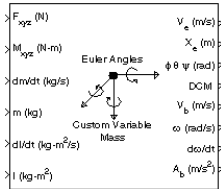Simple Variable Mass 3DoF (Body Axes)

Purpose   Implement an Euler angle representation of six-degrees-of-freedom equations of motion
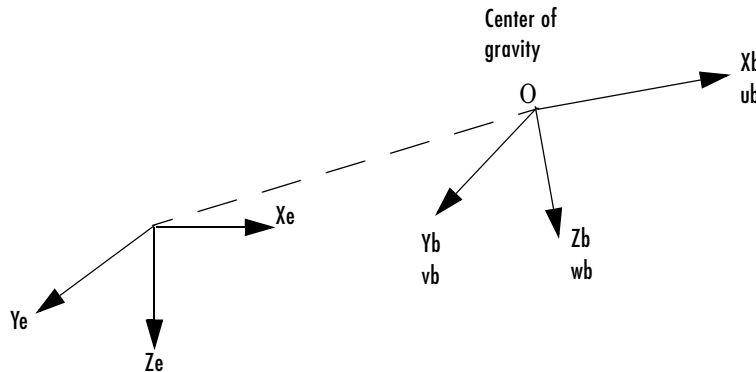
Library   Equations of Motion/6DoF

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 an Earth-fixed 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 Earth-fixed reference frame is considered inertial, a simplification that allows the forces due to the Earth's motion relative to a star-fixed reference system to be neglected.

Earth-fixed reference frame

The translational motion of the body-fixed coordinate frame is given below, where the applied forces $[\mathrm{F}_x\ \mathrm{F}_y\ \mathrm{F}_z]^\mathrm{T}$ are in the body-fixed frame.

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

Custom Variable Mass 6DoF (Euler Angles)

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

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

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

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

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

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

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

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

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

Dialog Box

Block Parameters: Custom Variable Mass 6DoF (Euler Angles)

6DoF EoM (Body Axis) (mask) (link)

Integrate the six-degrees-of-freedom equations of motion using an Euler angle representation for the orientation of the body in space.

Parameters

Units: Metric (MKS)

Mass type: Custom Variable

Representation: Euler Angles

Initial position in inertial axes [Xe,Ye,Ze]:

[0 0 0]

Initial velocity in body axes [U,v,w]:

[0 0 0]

Initial Euler orientation [roll, pitch, yaw]:

[0 0 0]

Initial body rotation rates [p,q,r]:

[0 0 0]

OK   Cancel   Help   Apply

Custom Variable Mass 6DoF (Euler Angles)

Units

Specifies the input and output 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:

Mass	Description
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:

Mass	Description
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 Earth-fixed reference frame.

Initial velocity in body axes

The three-element vector for the initial velocity in the body-fixed coordinate frame.

Initial Euler rotation

The three-element vector for the initial Euler rotation angles [roll, pitch, yaw], in radians.

Initial body rotation rates

The three-element vector for the initial body-fixed angular rates, in radians per second.

Inputs and Outputs

The first input to the block is a vector containing the three applied forces.

The second input is a vector containing the three applied moments.

The third input is a scalar containing the rate of change of mass.

The fourth input is a scalar containing the mass

The fifth input is a 3-by-3 matrix for the rate of change of inertia tensor matrix.

The sixth input is a 3-by-3 matrix for the inertia tensor matrix.

The first output is a three-element vector containing the velocity in the Earth-fixed reference frame.

The second output is a three-element vector containing the position in the Earth-fixed reference frame.

The third output is a three-element vector containing the Euler rotation angles [roll, pitch, yaw], in radians.

The fourth output is a 3-by-3 matrix for the coordinate transformation from Earth-fixed axes to body-fixed axes.

The fifth output is a three-element vector containing the velocity in the body-fixed frame.

Custom Variable Mass 6DoF (Euler Angles)

The sixth output is a three-element vector containing the angular rates in body-fixed axes, in radians per second.

The seventh output is a three-element vector containing the angular accelerations in body-fixed axes, in radians per second.

The eighth output is a three-element vector containing the accelerations in body-fixed axes.

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

References Mangiacasale, L., "Flight Mechanics of a u-Airplane with a MATLAB Simulink Helper," Edizioni Libreria CLUP, 1998.

See Also 6DoF (Euler Angles)

6DoF (Quaternion)

Custom Variable Mass 6DoF (Quaternion)

Simple Variable Mass 6DoF (Euler Angles)

Simple Variable Mass 6DoF (Quaternion)

Custom Variable Mass 6DoF (Quaternion)

Purpose

Implement a quaternion 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 Custom Variable Mass 6DoF (Euler Angles) block.
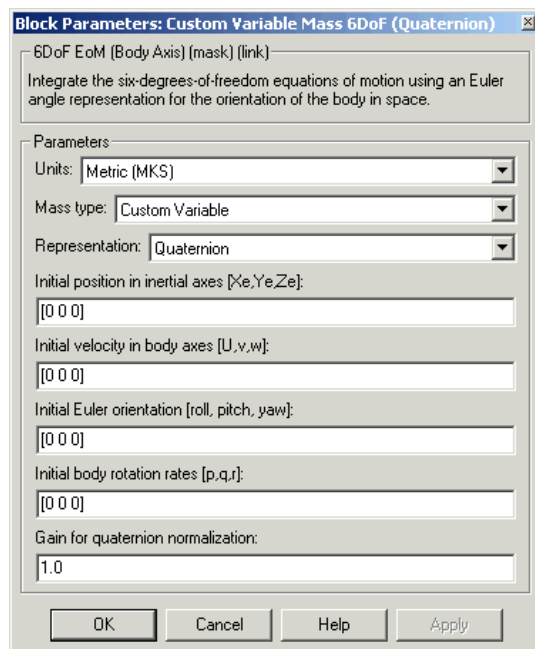
The integration of the rate of change of the quaternion vector is given below. The gain K drives the norm of the quaternion state vector to 1.0 should ε become nonzero. You must choose the value of this gain with care, because a large value improves the decay rate of the error in the norm, but also slows 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.

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

$$
\varepsilon = 1 - (q_0{}^2 + q_1{}^2 + q_3{}^2 + q_4{}^2)
$$

Custom Variable Mass 6DoF (Quaternion)

Dialog Box

Units

Specifies the input and output 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:

Mass	Description
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:

Mass	Description
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 Earth-fixed reference frame.

Initial velocity in body axes

The three-element vector for the initial velocity in the body-fixed coordinate frame.

Initial Euler rotation

The three-element vector for the initial Euler rotation angles [roll, pitch, yaw], in radians.

Initial body rotation rates

The three-element vector for the initial body-fixed angular rates, in radians per second.

Gain for quaternion normalization

> The gain to maintain the norm of the quaternion vector equal to 1.0.

Inputs and Outputs

The first input to the block is a vector containing the three applied forces.

The second input is a vector containing the three applied moments.

The third input is a scalar containing the rate of change of mass.

The fourth input is a scalar containing the mass

The fifth input is a 3-by-3 matrix for the rate of change of inertia tensor matrix.

The sixth input is a 3-by-3 matrix for the inertia tensor matrix.

The first output is a three-element vector containing the velocity in the Earth-fixed reference frame.

The second output is a three-element vector containing the position in the Earth-fixed reference frame.

The third output is a three-element vector containing the Euler rotation angles [roll, pitch, yaw], in radians.

The fourth output is a 3-by-3 matrix for the coordinate transformation from Earth-fixed axes to body-fixed axes.

The fifth output is a three-element vector containing the velocity in the body-fixed frame.

The sixth output is a three-element vector containing the angular rates in body-fixed axes, in radians per second.

The seventh output is a three-element vector containing the angular accelerations in body-fixed axes, in radians per second.

The eighth output is a three-element vector containing the accelerations in body-fixed axes.

Assumptions and Limitations

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

References

Mangiacasale, L., "Flight Mechanics of a u-Airplane with a MATLAB Simulink Helper," Edizioni Libreria CLUP, 1998.

See Also   6DoF (Euler Angles)

6DoF (Quaternion)

Custom Variable Mass 6DoF (Euler Angles)

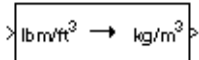Simple Variable Mass 6DoF (Euler Angles)

Simple Variable Mass 6DoF (Quaternion)

Density Conversion

Purpose   Convert from density units to desired density units

Library   Utilities/Unit Conversions

Description   The Density Conversion block computes the conversion factor from specified input density units to specified output density units and applies the conversion factor to the input signal.

The Density Conversion block icon displays the input and output units selected from the **Initial units** and the **Final units** lists.

Dialog Box
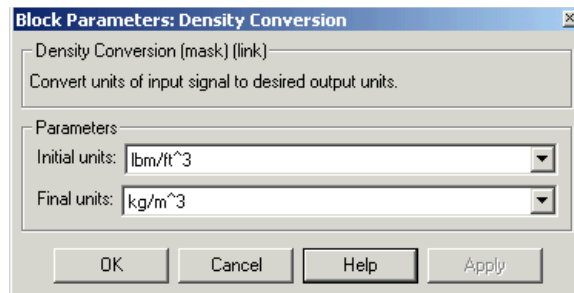
Initial units
> Specifies the input units.

Final units
> Specifies the output units.

The following conversion units are available:

lbm/ft^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   The input is density in initial density units.

The output is density in final density units.

See Also   Acceleration Conversion

Angle Conversion

Angular Acceleration Conversion

Angular Velocity Conversion

Force Conversion

Length Conversion

Mass Conversion

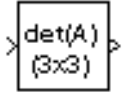Pressure Conversion

Temperature Conversion

Velocity Conversion

Determinant of 3x3 Matrix

Purpose   Compute the determinant for the input matrix

Library   Utilities/Math Operations

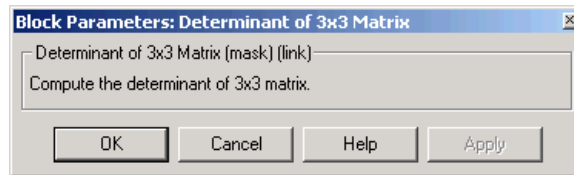**Description**   The Determinant of 3x3 Matrix block computes the determinant for the input matrix.

The input matrix has the form of

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

The determinant of the matrix has the form of

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

Dialog Box

Block Parameters: Determinant of 3x3 Matrix

Determinant of 3x3 Matrix (mask) (link)

Compute the determinant of 3x3 matrix.

| OK | Cancel | Help | Apply |

Inputs and Outputs   The input is a 3-by-3 matrix.

The output of the block is the determinant of input matrix.

See Also   Adjoint of 3x3 Matrix

Create 3x3 Matrix

Invert 3x3 Matrix

Purpose   Convert direction cosine matrix to Euler angles

Library   Utilities/Axes Transformations

Description   The Direction Cosine Matrix to Euler Angles block converts a 3-by-3 direction cosine matrix (DCM) into three Euler rotation angles. The DCM matrix performs the coordinate transformation of a vector in inertial axes (ox_0, oy_0, oz_0) into a vector in body axes (ox_3, oy_3, oz_3). The order of the axis rotations required to bring (ox_3, oy_3, oz_3) into coincidence with (ox_0, oy_0, oz_0) is first a rotation about ox_3 through the roll angle (ϕ) to axes (ox_2, oy_2, oz_2). Second a rotation about oy_2 through the pitch angle (θ) to axes (ox_1, oy_1, oz_1), and finally a rotation about oz_1 through the yaw angle (ψ) to axes (ox_0, oy_0, oz_0).

DCM2Eul

$$\begin{bmatrix} ox_3 \\ oy_3 \\ oz_3 \end{bmatrix} = DCM \begin{bmatrix} ox_0 \\ oy_0 \\ oz_0 \end{bmatrix}$$

$$\begin{bmatrix} ox_3 \\ oy_3 \\ oz_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} ox_0 \\ oy_0 \\ oz_0 \end{bmatrix}$$

Combining the three axis transformation matrices defines the following DCM.

$$DCM = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ (\sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi) & (\sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi) & \sin\phi\cos\theta \\ (\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi) & (\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi) & \cos\phi\cos\theta \end{bmatrix}$$

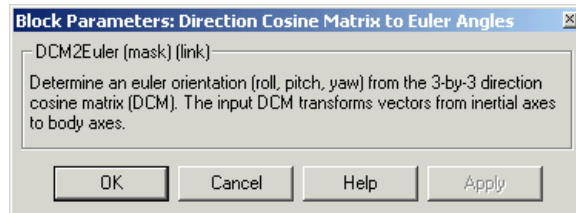To determine Euler angles from the DCM, the following equations are used:

$$\phi = \operatorname{atan}\left(\frac{DCM(2,3)}{DCM(3,3)}\right)$$
$$\theta = \operatorname{asin}(-DCM(1,3))$$
$$\psi = \operatorname{atan}\left(\frac{DCM(1,2)}{DCM(1,1)}\right)$$

Direction Cosine Matrix to Euler Angles

Dialog Box

Inputs and Outputs
The input is a 3-by-3 direction cosine matrix.

The output is a 3-by-1 vector of Euler angles.

Assumptions and Limitations
This implementation generates a pitch angle that lies between ±90 degrees, and roll and yaw angles that lie between ±180 degrees.

See Also
Direction Cosine Matrix to Quaternions

Euler Angles to Direction Cosine Matrix

Euler Angles to Quaternions

Quaternions to Direction Cosine Matrix

Quaternions to Euler Angles

Direction Cosine Matrix to Quaternions

Purpose   Convert direction cosine matrix to quaternion vector

Library   Utilities/Axes Transformations

Description

> DCM2Quat >
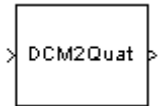
The Direction Cosine Matrix to Quaternions block transforms a 3-by-3 direction cosine matrix (DCM) into a four-element unit quaternion vector (q_0,q_1,q_2,q_3). The DCM performs the coordinate transformation of a vector in inertial axes to a vector in body axes.

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 is 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.

Dialog Box

Block Parameters: Direction Cosine Matrix to Quaternions ⊠

─ DCM2Quaternion (mask) (link) ──────────────────────

Determine the 4x1 quaternion orientation vector from a 3-by-3 direction cosine matrix (DCM). The input DCM transforms vectors from inertial axes to body axes.

──

| OK | Cancel | Help | Apply |
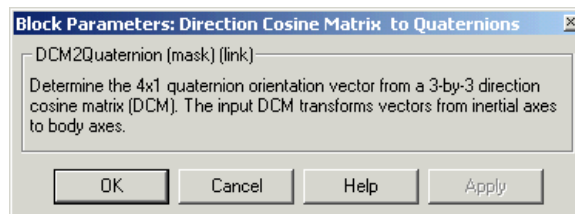
Inputs and Outputs

The input is a 3-by-3 direction cosine matrix.

The output is a 4-by-1 quaternion vector.

Direction Cosine Matrix to Quaternions

Purpose   Generate discrete wind gust

**Library**   Environment/Wind

Description   The Discrete Wind Gust Model block implements a wind gust of the standard
                   "1-cosine" shape. This block implements the mathematical representation in
                   the Military Specification MIL-F-8785C [1]. The gust is applied to each axis
                   individually, or to all three axes at once. The user specifies 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.
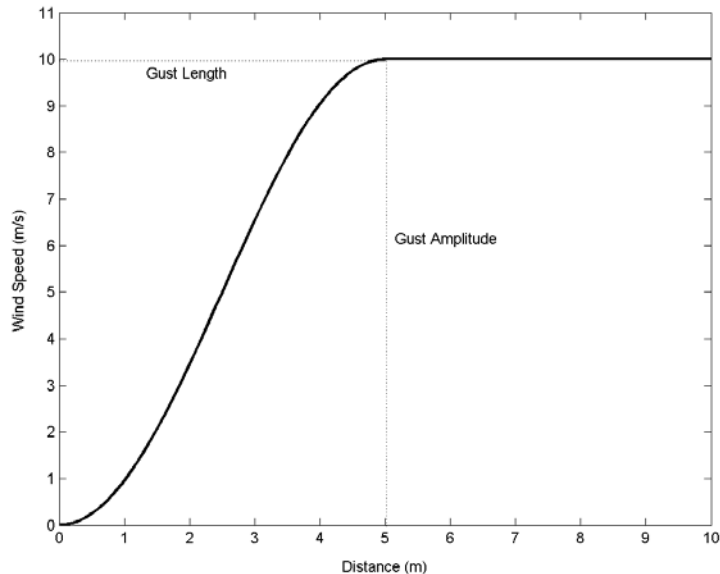
The discrete gust can be used singly or in multiples to assess airplane response
to large wind disturbances.

Discrete Wind Gust Model

The mathematical representation of the discrete gust is

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

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.

Dialog Box

Units

Define the units of wind gust.

	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

The input is airspeed in units selected.

The output is wind speed in units selected.

Examples

See Airframe in the `aeroblk_HL20` demo for an example of this block.

References

U.S. Military Specification MIL-F-8785C, 5 November 1980.
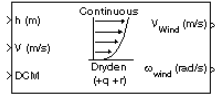
See Also

Dryden Wind Turbulence Model (Continuous), Wind Shear Model

Dryden Wind Turbulence Model (Continuous)

Purpose   Generate continuous wind turbulence with the Dryden velocity spectra

Library   Environment/Wind

Description

The Dryden Wind Turbulence Model (Continuous) block uses the Dryden spectral representation to add turbulence to the aerospace model by passing band-limited white noise through appropriate forming filters. This block implements the mathematical representation in the Military Specification MIL-F-8785C and Military Handbook MIL-HDBK-1797.

According to the military references, turbulence is a stochastic process defined by velocity spectra. For an aircraft flying at a speed V through a "frozen" turbulence field with a spatial frequency of 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**
Longitudinal		
$\Phi_u(\omega)$	$\dfrac{2\sigma_u^2 L_u}{\pi V} \cdot \dfrac{1}{1 + (L_u \frac{\omega}{V})^2}$	$\dfrac{2\sigma_u^2 L_u}{\pi V} \cdot \dfrac{1}{1 + (L_u \frac{\omega}{V})^2}$
$\Phi_{p_g}(\omega)$	$\dfrac{\sigma_w^2}{V L_w} \cdot \dfrac{0.8\left(\frac{\pi L_w}{4b}\right)^{\frac{1}{3}}}{1 + \left(\frac{4b\omega}{\pi V}\right)^2}$	$\dfrac{\sigma_w^2}{2V L_w} \cdot \dfrac{0.8\left(\frac{2\pi L_w}{4b}\right)^{\frac{1}{3}}}{1 + \left(\frac{4bw}{\pi V}\right)^2}$

Dryden Wind Turbulence Model (Continuous)

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

The variable b represents the aircraft wingspan. The variables L_u, L_v, L_w represent the turbulence scale lengths. The variables $\sigma_u, \sigma_v, \sigma_w$ represent the turbulence intensities.

Dryden Wind Turbulence Model (Continuous)

The spectral density definitions of turbulence angular rates are defined in the specifications as three variations, which are displayed in the following table:

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

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

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

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

Keep in mind that the longitudinal turbulence angular rate spectrum, $\Phi_{p_g}(\omega)$, is a rational function. The rational function is derived from curve-fitting a complex algebraic function, not the vertical turbulence velocity spectrum, $\Phi_w(\omega)$, multiplied by a scale factor. Because the turbulence angular rate spectra contribute less to the aircraft gust response than the turbulence velocity spectra, it may explain the variations in their definitions.

The variations lead to the following combinations of vertical and lateral turbulence angular rate spectra:

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 passed through forming filters. The forming filters are derived from the spectral square roots of the spectrum equations.

The following table displays the transfer functions:

	MIL-F-8785C	**MIL-HDBK-1797**

Longitudinal

$H_u(s)$

$$\sigma_u \sqrt{\frac{2L_u}{\pi V}} \frac{1}{1 + \frac{L_u}{V}s} \qquad \qquad \sigma_u \sqrt{\frac{2L_u}{\pi V}} \frac{1}{1 + \frac{L_u}{V}s}$$

$H_p(s)$

$$\sigma_w \sqrt{\frac{0.8}{V}} \frac{\left(\frac{\pi}{(4b)}\right)^{1/6}}{L_w^{1/3}\left(1 + \left(\frac{4b}{\pi V}\right)s\right)} \qquad \sigma_w \sqrt{\frac{0.8}{V}} \frac{\left(\frac{\pi}{(4b)}\right)^{1/6}}{(2L_w)^{1/3}\left(1 + \left(\frac{4b}{\pi V}\right)s\right)}$$

Lateral

$H_v(s)$

$$\sigma_v \sqrt{\frac{L_v}{\pi V}} \frac{1 + \frac{\sqrt{3}L_v}{V}s}{(1 + \frac{L_v}{V}s)^2} \qquad \sigma_v \sqrt{\frac{2L_v}{\pi V}} \frac{1 + \frac{2\sqrt{3}L_v}{V}s}{\left(1 + \frac{2L_v}{V}s\right)^2}$$

$H_r(s)$

$$\frac{\mp\frac{s}{V}}{\left(1 + \left(\frac{3b}{\pi V}\right)s\right)} \cdot H_v(s) \qquad \frac{\mp\frac{s}{V}}{\left(1 + \left(\frac{3b}{\pi V}\right)s\right)} \cdot H_v(s)$$

Vertical

$H_w(s)$

$$\sigma_w \sqrt{\frac{L_w}{\pi V}} \frac{1 + \frac{\sqrt{3}L_w}{V}s}{(1 + \frac{L_w}{V}s)^2} \qquad \sigma_w \sqrt{\frac{2L_w}{\pi V}} \frac{1 + \frac{2\sqrt{3}L_w}{V}s}{\left(1 + \frac{2L_w}{V}s\right)^2}$$

$H_q(s)$

$$\frac{\pm\frac{s}{V}}{\left(1 + \left(\frac{4b}{\pi V}\right)s\right)} \cdot H_w(s) \qquad \frac{\pm\frac{s}{V}}{\left(1 + \left(\frac{4b}{\pi V}\right)s\right)} \cdot H_w(s)$$

Dryden Wind Turbulence Model (Continuous)

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 < 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:

<div align="center">

MIL-F-8785C   **MIL-HDBK-1797**

</div>

$$L_w = h \qquad\qquad\qquad 2L_w = h$$

$$L_u = L_v = \frac{h}{(0.177 + 0.000823h)^{1.2}} \qquad L_u = 2L_v = \frac{h}{(0.177 + 0.000823h)^{1.2}}$$

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.

$$\sigma_w = 0.1 W_{20}$$

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

The turbulence axes orientation in this region is defined as follows:

- Longitudinal turbulence velocity, u_g, aligned along the horizontal relative mean wind vector
- Vertical turbulence velocity, w_g, aligned with vertical

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

Medium/High Altitudes (Altitude > 2000 feet)

For medium to high altitudes the turbulence scale lengths and intensities are based on the assumption that the turbulence is isotropic. In the military references, the scale lengths are represented by the following equations:

MIL-F-8785C	**MIL-HDBK-1797**
$L_u = L_v = L_w = 1750\,ft$	$L_u = 2L_v = 2L_w = 1750\,ft$

The turbulence intensities are determined from a lookup table that provides the turbulence intensity as a function of altitude and the probability of the turbulence intensity being exceeded. The relationship of the turbulence intensities is represented in the following equation.

$$\sigma_u = \sigma_v = \sigma_w$$

Dryden Wind Turbulence Model (Continuous)

The turbulence axes orientation in this region is defined as being aligned with the body coordinates.

Medium/High Altitude Turbulence Intensities (Probability of Exceedance)

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.

Dialog Box

Dryden Wind Turbulence Model (Continuous)

Units

Define the units of wind speed due to the turbulence.

	Wind Velocity	Altitude	Air Speed
`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.

Model	Description
Continuous Von Kármán (+q -r)	Use continuous representation of Von Kármán velocity spectra with positive vertical and negative lateral angular rates spectra.
Continuous Von Kármán (+q +r)	Use continuous representation of Von Kármán velocity spectra with positive vertical and lateral angular rates spectra.
Continuous Von Kármán (-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.

Model	Description
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

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.

Dryden Wind Turbulence Model (Continuous)

Scale length at medium/high altitudes (m)

The turbulence scale length above 2000 feet is assumed constant, and from the military references, a figure of 1750 feet is recommended for the longitudinal turbulence scale length of the Dryden spectra.

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

Wingspan

The wingspan is required in the calculation of the turbulence on the angular rates.

Band-limited noise sample time (sec)

The sample time at which the unit variance white noise signal is generated.

Noise seeds

There are four random numbers required to generate the turbulence signals, one for each of the three velocity components and one for the roll rate. The turbulences on the pitch and yaw angular rates are based on further shaping of the outputs from the shaping filters for the vertical and lateral velocities.

Turbulence on

Selecting the check box generates the turbulence signals.

Inputs and Outputs

The first input is altitude, in units selected.

The second input is aircraft speed, in units selected.

The third input is a direction cosine matrix.

The first output is a three-element signal containing the turbulence velocities, in the selected units.

The second output is a three-element signal containing 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)

Examples

See the Airframe subsystem in the `aeroblk_HL20` demo for an example of this block.

References

U.S. Military Handbook MIL-HDBK-1797, 19 December 1997.

U.S. Military Specification MIL-F-8785C, 5 November 1980.

Chalk, C., Neal, P., Harris, T., Pritchard, F., Woodcock, R., "Background Information and User Guide for MIL-F-8785B(ASG), 'Military Specification-Flying Qualities of Piloted Airplanes'," AD869856, Cornell Aeronautical Laboratory, August 1969.

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

Ly, U., Chan, Y., "Time-Domain Computation of Aircraft Gust Covariance Matrices," AIAA Paper 80-1615, Atmospheric Flight Mechanics Conference, Danvers, 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.

Dryden Wind Turbulence Model (Continuous)

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
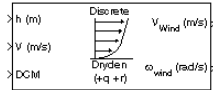
Dryden Wind Turbulence Model (Discrete)

Discrete Wind Gust Model

Wind Shear Model

Von Karman Wind Turbulence Model (Continuous)

Dryden Wind Turbulence Model (Discrete)

Purpose   Generate continuous wind turbulence with the Dryden velocity spectra

Library   Environment/Wind

Description

The Dryden Wind Turbulence Model (Discrete) block uses the Dryden spectral representation to add turbulence to the aerospace model by using band-limited white noise with appropriate digital filter finite difference equations. This block implements the mathematical representation in the Military Specification MIL-F-8785C and Military Handbook MIL-HDBK-1797.

According to the military references, turbulence is a stochastic process defined by velocity spectra. For an aircraft flying at a speed V through a "frozen" turbulence field with a spatial frequency of 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**
Longitudinal		
$\Phi_u(\omega)$	$\dfrac{2\sigma_u^2 L_u}{\pi V} \cdot \dfrac{1}{1 + (L_u \frac{\omega}{V})^2}$	$\dfrac{2\sigma_u^2 L_u}{\pi V} \cdot \dfrac{1}{1 + (L_u \frac{\omega}{V})^2}$
$\Phi_p(\omega)$	$\dfrac{\sigma_w^2}{V L_w} \cdot \dfrac{0.8\left(\frac{\pi L_w}{4b}\right)^{\frac{1}{3}}}{1 + \left(\frac{4b\omega}{\pi V}\right)^2}$	$\dfrac{\sigma_w^2}{2V L_w} \cdot \dfrac{0.8\left(\frac{2\pi L_w}{4b}\right)^{\frac{1}{3}}}{1 + \left(\frac{4bw}{\pi V}\right)^2}$

Dryden Wind Turbulence Model (Discrete)

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

The variable b represents the aircraft wingspan. The variables L_u, L_v, L_w represent the turbulence scale lengths. The variables $\sigma_u, \sigma_v, \sigma_w$ represent the turbulence intensities.

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_q) 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.

Dryden Wind Turbulence Model (Discrete)

The following table displays the digital filter finite difference equations:

	MIL-F-8785C	**MIL-HDBK-1797**

Longitudinal

u_g
$$\left(1 - \frac{V}{L_u}T\right)u_g + \sqrt{2\frac{V}{L_u}T}\frac{\sigma_u}{\sigma_\eta}\eta_1 \qquad \left(1 - \frac{V}{L_u}T\right)u_g + \sqrt{2\frac{V}{L_u}T}\frac{\sigma_u}{\sigma_\eta}\eta_1$$

p_g
$$\left(1 - \frac{2.6}{\sqrt{L_w b}}T\right)p_g + \qquad \left(1 - \frac{2.6}{\sqrt{2L_w b}}T\right)p_g +$$

$$\sqrt{2\frac{2.6}{\sqrt{L_w b}}T}\frac{\frac{0.95}{\sqrt[3]{2L_w b^2}}\sigma_w}{\sigma_\eta}\eta_4 \qquad \sqrt{2\frac{2.6}{\sqrt{2L_w b}}T}\frac{\frac{1.9}{\sqrt{2L_w b}}\sigma_w}{\sigma_\eta}\eta_4$$

Lateral

v_g
$$\left(1 - \frac{V}{L_u}T\right)v_g + \sqrt{2\frac{V}{L_u}T}\frac{\sigma_v}{\sigma_\eta}\eta_2 \qquad \left(1 - \frac{V}{L_u}T\right)v_g + \sqrt{2\frac{V}{L_u}T}\frac{\sigma_v}{\sigma_\eta}\eta_2$$

r_g
$$\left(1 - \frac{\pi V}{3b}T\right)r_g \mp \frac{\pi}{3b}(v_g - v_{g_{past}}) \qquad \left(1 - \frac{\pi V}{3b}T\right)r_g \mp \frac{\pi}{3b}(v_g - v_{g_{past}})$$

Vertical

w_g
$$\left(1 - \frac{V}{L_u}T\right)w_g + \sqrt{2\frac{V}{L_u}T}\frac{\sigma_w}{\sigma_\eta}\eta_3 \qquad \left(1 - \frac{V}{L_u}T\right)w_g + \sqrt{2\frac{V}{L_u}T}\frac{\sigma_w}{\sigma_\eta}\eta_3$$

q_g
$$\left(1 - \frac{\pi V}{4b}T\right)q_g \pm \frac{\pi}{4b}(w_g - w_{g_{past}}) \qquad \left(1 - \frac{\pi V}{4b}T\right)q_g \pm \frac{\pi}{4b}(w_g - w_{g_{past}})$$

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

Low-Altitude Model (Altitude < 1000 feet)

According to the military references, the turbulence scale lengths at low altitudes, where h is the altitude in feet, are represented in the following table:

MIL-F-8785C   **MIL-HDBK-1797**

$$L_w = h \qquad\qquad\qquad\qquad 2L_w = h$$

$$L_u = L_v = \frac{h}{(0.177 + 0.000823h)^{1.2}} \qquad L_u = 2L_v = \frac{h}{(0.177 + 0.000823h)^{1.2}}$$

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.

Dryden Wind Turbulence Model (Discrete)

Medium/High Altitudes (Altitude > 2000 feet)

For medium to high altitudes the turbulence scale lengths and intensities are based on the assumption that the turbulence is isotropic. In the military references, the scale lengths are represented by the following equations:

MIL-F-8785C   **MIL-HDBK-1797**

$$L_u = L_v = L_w = 1750 \, ft \qquad\qquad L_u = 2L_v = 2L_w = 1750 \, ft$$

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.

Dialog Box

Dryden Wind Turbulence Model (Discrete)

Units

Define the units of wind speed due to the turbulence.

	Wind Velocity	Altitude	Air Speed
`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:

Model	Description
Continuous Von Kármán (+q -r)	Use continuous representation of Von Kármán velocity spectra with positive vertical and negative lateral angular rates spectra.
Continuous Von Kármán (+q +r)	Use continuous representation of Von Kármán velocity spectra with positive vertical and lateral angular rates spectra.
Continuous Von Kármán (-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.

Model	Description
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.

Dryden Wind Turbulence Model (Discrete)

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

The first input is altitude, in units selected.

The second input is aircraft speed, in units selected.

The third input is a direction cosine matrix.

The first output is a three-element signal containing the turbulence velocities, in the selected units.

The second output is a three-element signal containing 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-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.

Dryden Wind Turbulence Model (Discrete)

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)
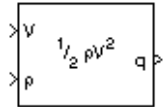
Von Karman Wind Turbulence Model (Continuous)

Dryden Wind Turbulence Model (Discrete)

Discrete Wind Gust Model

Purpose   Compute dynamic pressure using velocity and air density

Library   Flight Parameters

Description   The Dynamic Pressure block computes dynamic pressure.

Dynamic pressure is defined as

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

where ρ is air density and V is velocity.

Dialog Box
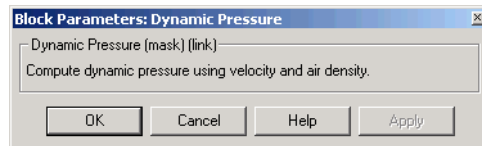
Inputs and Outputs   The first input is velocity vector.

The second input is air density.

The output of the block is dynamic pressure.

Examples   See the Airframe subsystem in the `aeroblk_HL20` demo for an example of this block.

See Also   Aerodynamic Forces and Moments

Mach Number

Estimate Center of Gravity

Purpose   Calculate the center of gravity location

Library   Mass Properties

Description   The Estimate Center of Gravity block calculates the center of gravity location
                     and the rate of change of the center of gravity.

Linear interpolation is used to estimate the location of center of gravity as a
function of mass. The rate of change of center of gravity is a linear function of
rate of change of mass.
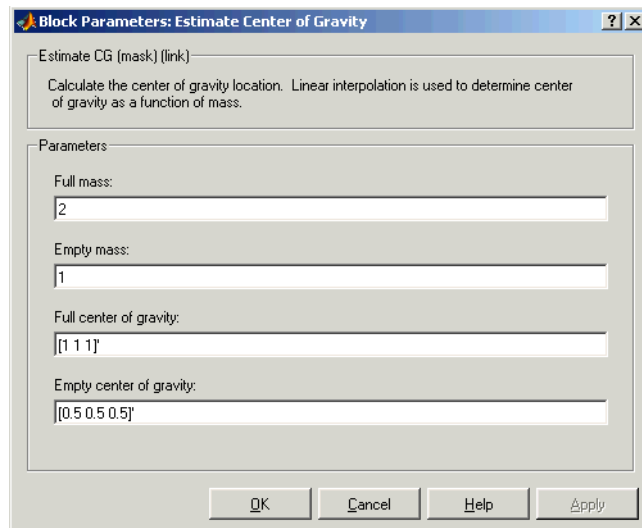
Dialog Box

Full mass
Specifies the gross mass of the craft.

Empty mass
Specifies the empty mass of the craft.

Full center of gravity
Specifies the center of gravity at gross mass of the craft.

Empty center of gravity

Specifies the center of gravity at empty mass of the craft.

Inputs and Outputs

The first input is the mass.

The second input is the rate of change of mass.

The first output is the center of gravity location.

The second output is the rate of change of center of gravity location.

See Also

Aerodynamic Forces and Moments

Estimate Inertia Tensor

Moments About CG Due to Forces

Estimate Inertia Tensor

Purpose

Calculate the inertia tensor

Library

Mass Properties

Description
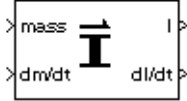
The Estimate Inertia Tensor block calculates the inertia tensor and the rate of change of the inertia tensor.

Linear interpolation is used to estimate the inertia tensor as a function of mass. The rate of change of the inertia tensor is a linear function of rate of change of mass.
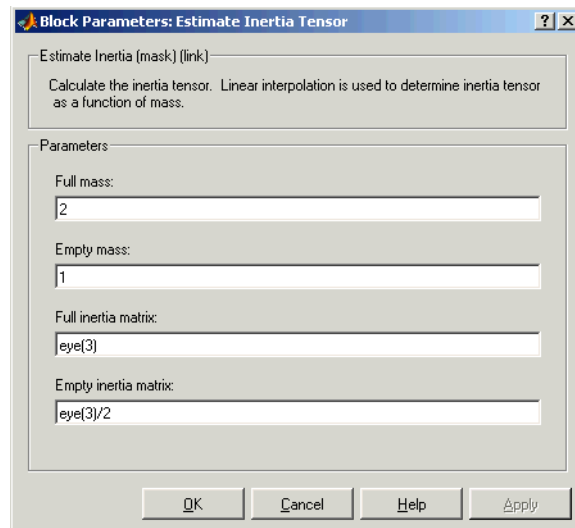
Dialog Box

Full mass

Specifies the gross mass of the craft.

Empty mass

Specifies the empty mass of the craft.

Full inertia matrix

Specifies the inertia tensor at gross mass of the craft.

Empty inertia matrix

Specifies the inertia tensor at empty mass of the craft.

Inputs and Outputs

The first input is mass.

The second input is rate of change of mass.

The first output is inertia tensor.

The second output is rate of change of inertia tensor.

See Also

Estimate Center of Gravity

Symmetric Inertia Tensor

Euler Angles to Direction Cosine Matrix

Purpose   Convert Euler angles to direction cosine matrix

Library   Utilities/Axes Transformations

Description   The Euler Angles to Direction Cosine Matrix block converts the three Euler
rotation angles into a 3-by-3 direction cosine matrix (DCM). The DCM matrix
performs the coordinate transformation of a vector in inertial axes
(ox_0, oy_0, oz_0) into a vector in body axes (ox_3, oy_3, oz_3). The order of the axis
rotations required to bring (ox_3, oy_3, oz_3) into coincidence with (ox_0, oy_0, oz_0)
is first a rotation about ox_3 through the roll angle (ϕ) to axes (ox_2, oy_2, oz_2).
Second a rotation about oy_2 through the pitch angle (θ) to axes (ox_1, oy_1, oz_1),
and finally a rotation about oz_1 through the yaw angle (ψ) to
axes (ox_0, oy_0, oz_0).

$$
\begin{bmatrix} ox_3 \\ oy_3 \\ oz_3 \end{bmatrix} = DCM \begin{bmatrix} ox_0 \\ oy_0 \\ oz_0 \end{bmatrix}
$$

$$
\begin{bmatrix} ox_3 \\ oy_3 \\ oz_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} ox_0 \\ oy_0 \\ oz_0 \end{bmatrix}
$$

Combining the three axis transformation matrices defines the following DCM.

$$
DCM = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ (\sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi) & (\sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi) & \sin\phi\cos\theta \\ (\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi) & (\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi) & \cos\phi\cos\theta \end{bmatrix}
$$

Dialog Box

```
Block Parameters: Euler Angles to  Direction Cosine Matrix      ⊠
─Euler2DCM (mask) (link)──────────────────────────────────
Determine the 3-by-3 direction cosine matrix (DCM) from an Euler
orientation (roll, pitch, yaw). The output DCM transforms vectors from
inertial axes to body axes.

      ┌─────────┐  ┌─────────┐  ┌─────────┐  ┌─────────┐
      │   OK    │  │ Cancel  │  │  Help   │  │  Apply  │
      └─────────┘  └─────────┘  └─────────┘  └─────────┘
```

Inputs and Outputs

The input is a 3-by-1 vector of Euler angles.

The output is a 3-by-3 direction cosine matrix.

See Also

Direction Cosine Matrix to Euler Angles

Direction Cosine Matrix to Quaternions

Euler Angles to Quaternions

Quaternions to Direction Cosine Matrix

Quaternions to Euler Angles

Euler Angles to Quaternions

Purpose   Convert Euler angles to a quaternion vector

Library   Utilities/Axes Transformations
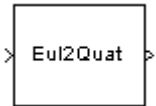
Description   The Euler Angles to Quaternions block converts the rotation described by the three Euler angles (roll, pitch, yaw) into the four-element quaternion vector (q_0,q_1,q_2,q_3).

Eul2Quat

A quaternion vector represents a rotation about a unit vector ($\mu_x\ \mu_y\ \mu_z$) through an angle θ. A unit quaternion itself has unit magnitude, and can be written in the following vector format.

$$q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos(\theta/2) \\ \sin(\theta/2)\mu_x \\ \sin(\theta/2)\mu_y \\ \sin(\theta/2)\mu_z \end{bmatrix}$$

An alternative representation of a quaternion is as a complex number,

$$q = q_0 + \boldsymbol{i}q_1 + \boldsymbol{j}q_2 + \boldsymbol{k}q_3$$

where, for the purposes of multiplication,

$$\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = -1$$

$$\boldsymbol{ij} = -\boldsymbol{ji} = \boldsymbol{k}$$

$$\boldsymbol{jk} = -\boldsymbol{kj} = \boldsymbol{i},$$
$$\boldsymbol{ki} = -\boldsymbol{ik} = \boldsymbol{j}$$

The benefit of representing the quaternion in this way is the ease with which the quaternion product can represent the resulting transformation after two or more rotations. The quaternion to represent the rotation through the three Euler angles is given below.

$$q = q_\phi q_\theta q_\psi = \left(\cos\!\left(\frac{\phi}{2}\right) - \boldsymbol{i}\sin\!\left(\frac{\phi}{2}\right) \right)\!\left(\cos\!\left(\frac{\theta}{2}\right) - \boldsymbol{j}\sin\!\left(\frac{\theta}{2}\right) \right)\!\left(\cos\!\left(\frac{\psi}{2}\right) - \boldsymbol{k}\sin\!\left(\frac{\psi}{2}\right) \right)$$

Expanding the preceding representation gives the four quaternion elements following.

$$
\begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos\left(\dfrac{\phi}{2}\right)\cos\left(\dfrac{\theta}{2}\right)\cos\left(\dfrac{\psi}{2}\right) + \sin\left(\dfrac{\phi}{2}\right)\sin\left(\dfrac{\theta}{2}\right)\sin\left(\dfrac{\psi}{2}\right) \\ \sin\left(\dfrac{\phi}{2}\right)\cos\left(\dfrac{\theta}{2}\right)\cos\left(\dfrac{\psi}{2}\right) - \cos\left(\dfrac{\phi}{2}\right)\sin\left(\dfrac{\theta}{2}\right)\sin\left(\dfrac{\psi}{2}\right) \\ \cos\left(\dfrac{\phi}{2}\right)\sin\left(\dfrac{\theta}{2}\right)\cos\left(\dfrac{\psi}{2}\right) + \sin\left(\dfrac{\phi}{2}\right)\cos\left(\dfrac{\theta}{2}\right)\sin\left(\dfrac{\psi}{2}\right) \\ \cos\left(\dfrac{\phi}{2}\right)\cos\left(\dfrac{\theta}{2}\right)\sin\left(\dfrac{\psi}{2}\right) - \sin\left(\dfrac{\phi}{2}\right)\sin\left(\dfrac{\theta}{2}\right)\cos\left(\dfrac{\psi}{2}\right) \end{bmatrix}
$$

Dialog Box

Block Parameters: Euler Angles to Quaternions

Euler2Quaternion (mask) (link)

Calculate quaternion (q0,q1,q2,q3) from Euler angles (roll, pitch, yaw)

| OK | Cancel | Help | Apply |
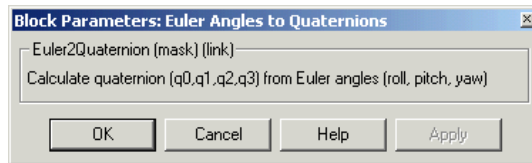
Inputs and Outputs

The input is a 3-by-1 vector of Euler angles.

The output is a 4-by-1 quaternion vector.

See Also

Direction Cosine Matrix to Euler Angles

Direction Cosine Matrix to Quaternions

Euler Angles to Direction Cosine Matrix

Quaternions to Direction Cosine Matrix

Quaternions to Euler Angles

Force Conversion

Purpose   Convert from force units to desired force units

Library   Utilities/Axes Transformations

Description The Force Conversion block computes the conversion factor from specified
input force units to specified output force units and applies the conversion
factor to the input signal.
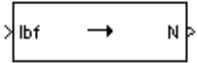
The Force Conversion block icon displays the input and output units selected
from the **Initial units** and the **Final units** lists.

Dialog Box
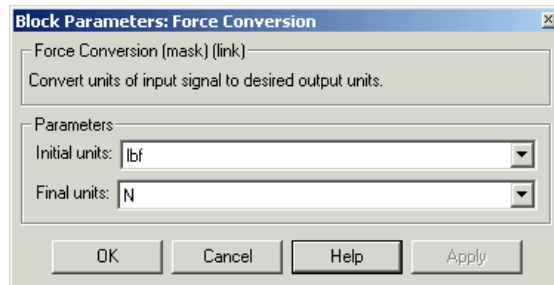
Initial units
Specifies the input units.

Final units
Specifies the output units.

The following conversion units are available:

lbf	Pound force
N	Newtons

**Inputs and
Outputs**   The input is force in initial force units.

The output is force in final force units.

See Also   Acceleration Conversion

Angle Conversion

Angular Acceleration Conversion

Angular Velocity Conversion

Density Conversion

Length Conversion

Mass Conversion

Pressure Conversion

Temperature Conversion

Velocity Conversion

Gain Scheduled Lead-Lag

Purpose   Implement a 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
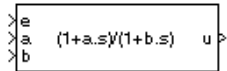
$$u = \frac{1 + as}{1 + bs}e$$

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 Look-Up Table (n-D) Simulink block.

Dialog Box
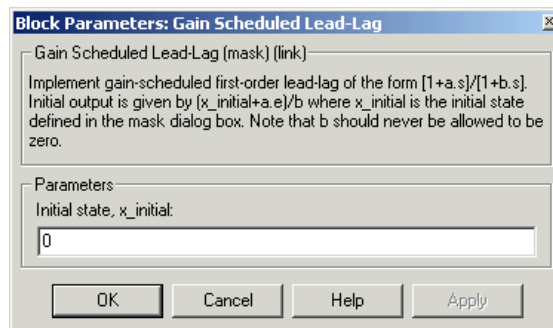
Initial state, x_initial

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

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

Inputs and Outputs

The first input is the filter input.

The second input is the numerator coefficient.

The third input is the denominator coefficient.

The output is the filter output

Purpose   Transform horizontal wind into body-axes coordinates

Library   Environment/Wind

Description   The Horizontal Wind Model block computes the wind velocity in body-axes coordinates.

The wind is specified by wind speed and wind direction in Earth axes. The speed and direction can be constant or variable over time. The direction of the wind is in degrees clockwise from the direction of the Earth x-axis (north). The wind direction is defined as the direction from which the wind is coming. Using the direction cosine matrix (DCM), the wind velocities are transformed into body-axes coordinates.

Dialog Box
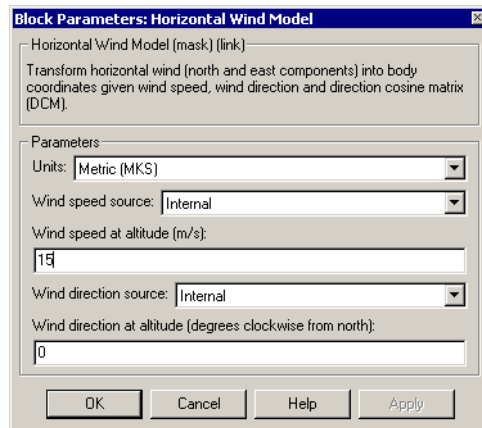
Units

Specifies the input and output units:

	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

Horizontal Wind Model

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

The first input is direction cosine matrix.

The second optional input is the wind speed in selected units.

The third optional input is the wind direction in degrees.

The output of the block is 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

Wind Shear Model

Purpose   Calculate equivalent airspeed (EAS), calibrated airspeed (CAS), or true airspeed (TAS) from each other

Library   Flight Parameters

Description

> TAS (m/s)
> a (m/s)   CAS (m/s) >
> P_o (Pa)

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*.

Ideal Airspeed Correction

Dialog Box
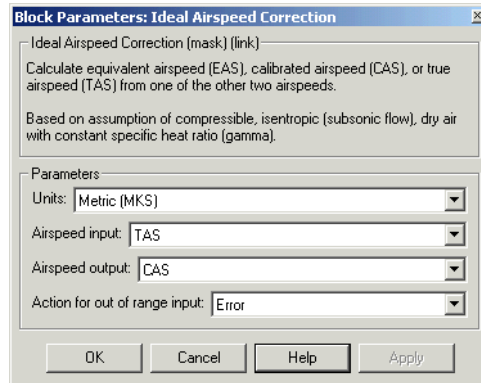
Units

Specifies the input and output 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

The first input is the selected airspeed in the selected units.

The second input is the speed of sound in the selected units.

The third input is the static pressure in the selected units.

The output of the block is 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.

Incidence & Airspeed

Purpose   Calculate incidence and air speed

Library   Flight Parameters
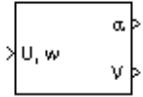
Description   The Incidence & Airspeed block supports the 3DoF equations of motion model by calculating the angle between the velocity vector and the body, and also the total air speed from the velocity components in the body-fixed coordinate frame.

$$\alpha = \text{atan}\left(\frac{w}{u}\right)$$

$$V = \sqrt{u^2 + w^2}$$

Dialog Box
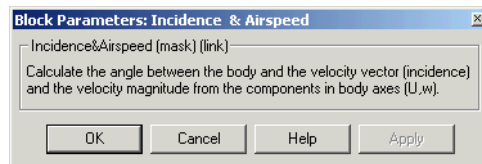
Inputs and Outputs   The input to the block is the two-element vector containing the velocity of the body resolved into the body-fixed coordinate frame.

The first output of the block is the incidence angle, in radians.

The second output is the air speed of the body.

Examples   See the `aeroblk_guidance` model and the `aero_guidance_airframe` model for examples of this block.

See Also   3DoF (Body Axes)

Incidence, Sideslip & Airspeed

Purpose   Calculate incidence, sideslip, and air speed

Library   Flight Parameters
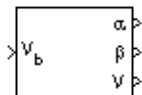
Description   The Incidence, Sideslip & Airspeed block supports the 6DoF (Euler Angles) and
6DoF (Quaternion) models by calculating the angles between the velocity
vector and the body, and also the total air speed from the velocity components
in the body-fixed coordinate frame.

$$\alpha = \operatorname{atan}\left(\frac{w}{u}\right)$$

$$\beta = \operatorname{asin}\left(\frac{v}{V}\right)$$

$$V = \sqrt{u^2 + v^2 + w^2}$$

Dialog Box

Block Parameters: Incidence, Sideslip & Airspeed

Incidence,Sideslip&Airspeed (mask) (link)

Calculate the angles between the body and the velocity vector (incidence
and sideslip), and the velocity magnitude from the components in body
axes (Vb).

| OK | Cancel | Help | Apply |
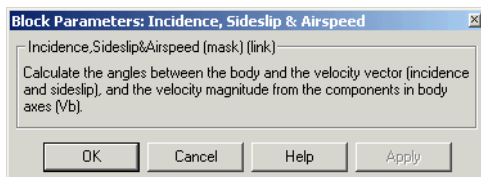
**Inputs and
Outputs**   The input to the block is the three-element vector containing the velocity of the
body resolved into the body-fixed coordinate frame.

The first output of the block is the incidence angle in radians.

The second output of the block is the sideslip angle in radians.

The third output is the air speed of the body.

Examples   See Airframe in the aeroblk_HL20 model for an example of this block.

See Also   6DoF (Euler Angles)

6DoF (Quaternion)

Incidence & Airspeed

Simple Variable Mass 6DoF (Euler Angles)

Incidence, Sideslip & Airspeed

Simple Variable Mass 6DoF (Quaternion)

Custom Variable Mass 6DoF (Euler Angles)

Custom Variable Mass 6DoF (Quaternion)

Purpose   Return an interpolated matrix for given input x

Library   GNC/Controls

Description   The Interpolate Matrix(x) block interpolates a one-dimensional array of matrices.

>|x Matrix(x) |>

This one-dimensional case assumes a matrix M is defined at a discrete number of values of an independent variable $x = [\, x_1\ x_2\ x_3\ ...\ x_i\ x_{i+1}\ ...\ x_n\,]$. 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

$$\lambda \;=\; (x - x_i)/(x_{i+1} - x_i)$$

Dialog Box

Block Parameters: Interpolate Matrix(x)

---MatrixSchedule-1D (mask) (link)---

Return an interpolated matrix for given input x. Input x must be from the Simulink PreLook-up Index Search block.

---Parameters---

Matrix to interpolate:

| matrix |

| OK | Cancel | Help | Apply |
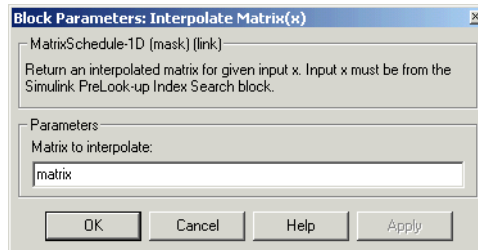
Matrix to interpolate

Matrix to be interpolated. It 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;
```

Inputs and Outputs   The first input is the first independent variable.

The output is the interpolated matrix.

Interpolate Matrix(x)

Assumptions and Limitations   This block must be driven from the Simulink PreLook-up Index Search block.

Examples   See the following Aerospace Blockset blocks: 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)

Purpose　　　　Return an interpolated matrix for given inputs x and y

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, $x = [\, x_1\, x_2\, x_3 \,...\, x_i\ x_{i+1} \,...\, x_n\,]$ and $y = [\, y_1\, y_2\, y_3 \,...\, y_j\, y_{j+1} \,...\, y_m]$. 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
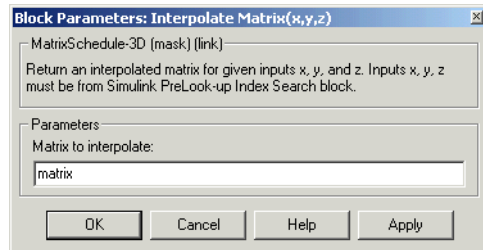
$$\lambda_x = (x - x_i)/(x_{i+1} - x_i)$$

and

$$\lambda_y = (y - y_j)/(y_{j+1} - y_j)$$

Dialog Box
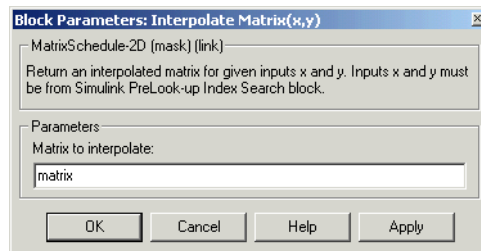
Matrix to interpolate

Matrix to be interpolated. It 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

Interpolate Matrix(x,y)

```
matrix(:,:,1,1) = A;
matrix(:,:,1,2) = B;
matrix(:,:,2,1) = C;
matrix(:,:,2,2) = D;
```

Inputs and Outputs   The first input is the first independent variable.

The second input is the second independent variable.

The output is the interpolated matrix.

Assumptions and Limitations   This block must be driven from the Simulink PreLookup Index Search block.

Examples   See the following Aerospace Blockset blocks: 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)

Purpose   Return an interpolated matrix for given inputs x, y, and z

Library   GNC/Controls

Description   The Interpolate Matrix(x,y,z) block interpolates a three-dimensional array of matrices.

This three-dimensional case assumes the matrix is defined as a function of three independent variables

$$x = [\, x_1\, x_2\, x_3 \dots x_i\ x_{i+1} \dots x_n\,], y = [\, y_1\, y_2\, y_3 \dots y_j\, y_{j+1} \dots y_m]$$
$$z = [\, z_1\, z_2\, z_3 \dots z_k\ z_{k+1} \dots z_p\,]$$

For given values of x, y, and z, eight matrices are interpolated. Then for

$$x_i < x < x_{i+1}, y_j < y < y_{j+1}$$
$$z_k < z < z_{k+1}$$

the output matrix is given by

$$
\begin{aligned}
&(1-\lambda_z)\,\{(1-\lambda_y)[(1-\lambda_x)M(x_i, y_j, z_k) + \lambda_x M(x_{i+1}, y_j, z_k)] + \\
&\qquad \lambda_y[(1-\lambda_x)M(x_i, y_{j+1}, z_k) + \lambda_x M(x_{i+1}, y_{j+1}, z_k)]\} \\
&+ \lambda_z\{(1-\lambda_y)[(1-\lambda_x)M(x_i, y_j, z_{k+1}) + \lambda_x M(x_{i+1}, y_j, z_{k+1})] + \\
&\qquad \lambda_y[(1-\lambda_x)M(x_i, y_{j+1}, z_{k+1}) + \lambda_x M(x_{i+1}, y_{j+1}, z_{k+1})]\}
\end{aligned}
$$

where the three interpolation fractions are denoted by

$$\lambda_x = (x - x_i)/(x_{i+1} - x_i)$$
$$\lambda_y = (y - y_j)/(y_{j+1} - y_j)$$
$$\lambda_z = (z - z_k)/(z_{k+1} - z_k)$$

In the three-dimensional case, the interpolation is carried out first on x, then y, and finally z.

Interpolate Matrix(x,y,z)

Dialog Box

Block Parameters: Interpolate Matrix(x,y,z)

MatrixSchedule-3D (mask) (link)

Return an interpolated matrix for given inputs x, y, and z. Inputs x, y, z must be from Simulink PreLook-up Index Search block.

Parameters

Matrix to interpolate:

matrix

| OK | Cancel | Help | Apply |

Matrix to interpolate

Matrix to be interpolated. It 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;
```

Inputs and Outputs

The first input is the first independent variable.

The second input is the second independent variable.

The third input is the third independent variable.

The output is the interpolated matrix.

Assumptions and Limitations

This block must be driven from the Simulink PreLookup Index Search block.

Examples   See the following Aerospace Blockset blocks: 3D Controller [A(v),B(v),C(v),D(v)], 3D Observer Form [A(v),B(v),C(v),F(v),H(v)], and 3D Self-Conditioned [A(v),B(v),C(v),D(v)].

See Also   Interpolate Matrix(x)

Interpolate Matrix(x,y)

Invert 3x3 Matrix

Purpose   Compute the inverse of 3-by-3 matrix using determinant formula

Library   Utilities/Math Operations

Description   The Invert 3x3 Matrix block computes the inverse of 3-by-3 matrix using determinant formula.

The inverse of the matrix is calculated by

$$inv(A) = \frac{adj(A)}{det(A)}$$

If the $det(A) = 0$, an error is thrown and the simulation will stop.

Dialog Box
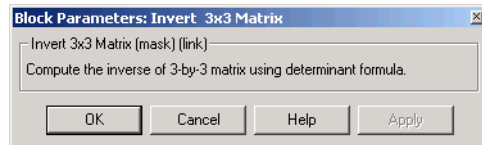
Inputs and Outputs   The input is a 3-by-3 matrix.

The output of the block is 3-by-3 matrix inverse of input matrix.

See Also   Adjoint of 3x3 Matrix

Create 3x3 Matrix

Determinant of 3x3 Matrix

Purpose   Implement the International Standard Atmosphere (ISA)

Library   Environment/Atmosphere
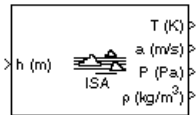
Description   The ISA Atmosphere Model block implements the mathematical
representation of the international standard atmosphere values for ambient
temperature, pressure, density, and speed of sound for the input geopotential
altitude.

The ISA Atmosphere Model block icon displays the input and output metric
units.

Dialog Box
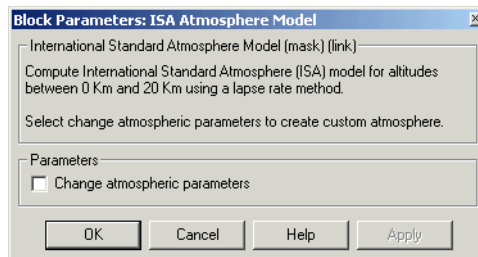
Change atmospheric parameters
    Select to customize various atmospheric parameters to be different from
    the ISA values.

**Inputs and
Outputs**   The input is geopotential height.

The four outputs are temperature, speed of sound, air pressure, and 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.

References   [1] U.S. Standard Atmosphere, 1976, U.S. Government Printing Office,
Washington, D.C.

See Also   COESA Atmosphere Model, Lapse Rate Model

Lapse Rate Model

Purpose   Implement lapse rate model for atmosphere

Library   Environment/Atmosphere

Description
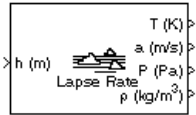
The Lapse Rate Model block implements the mathematical representation of the lapse rate atmospheric equations for ambient temperature, pressure, density, and speed of sound for the input geopotential altitude. You can customize this atmospheric model, described below, by specifying atmospheric properties in the block dialog.

The following equations define the troposphere

$$T = T_o - Lh$$

$$P = P_o \cdot \left(\frac{T}{T_o}\right)^{\frac{g}{LR}}$$

$$\rho = \rho_o \cdot \left(\frac{T}{T_o}\right)^{\frac{g}{LR} - 1}$$

$$a = \sqrt{\gamma RT}$$

The following equations define the tropopause (lower stratosphere)

$$T = 216.7^o K$$

$$P = P_o \cdot \left(\frac{T}{T_o}\right)^{\frac{g}{LR} - 1} \cdot e^{\frac{g}{RT}(11000 - h)}$$

$$\rho = \rho_o \cdot \left(\frac{T}{T_o}\right)^{\frac{g}{LR}} \cdot e^{\frac{g}{RT}(11000 - h)}$$

$$a = \sqrt{\gamma RT}$$

where:

T_0	Absolute temperature at mean sea level in degrees Kelvin
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
T	Absolute temperature at altitude h in degrees Kelvin
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 degrees Kelvin/m
R	Characteristic gas constant J/kg-degrees Kelvin
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.

Lapse Rate Model

Dialog Box

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).

Inputs and Outputs

The input is geopotential height.

The four outputs are temperature, speed of sound, air pressure, and 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.

References

[1] U.S. Standard Atmosphere, 1976, U.S. Government Printing Office, Washington, D.C.

See Also

COESA Atmosphere Model

ISA Atmosphere Model

Length Conversion

Purpose Convert from length units to desired length units

Library Utilities/Unit Conversions

Description The Length Conversion block computes the conversion factor from specified input length units to specified output length units and applies the conversion factor to the input signal.
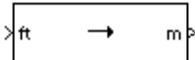
```
┌──────────────┐
│ ft    →    m │
└──────────────┘
```

The Length Conversion block icon displays the input and output units selected from the **Initial units** and the **Final units** lists.

Dialog Box
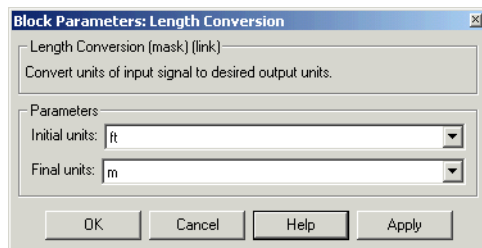
Initial units

 Specifies the input units.

Final units

 Specifies the output units.

The following conversion units are available:

m	Meters
ft	Feet
km	Kilometers
in	Inches
mi	Miles
naut mi	Nautical miles

Inputs and Outputs The input is length in initial length units.

The output is 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

Mach Number

**Purpose**   Compute Mach number using velocity and speed of sound

Library   Flight Parameters

Description   The Mach Number block computes Mach number.
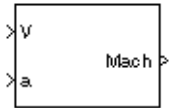
Mach number is defined as

$$Mach = \frac{\sqrt{V \cdot V}}{a}$$

where a is speed of sound and V is velocity vector.

Dialog Box
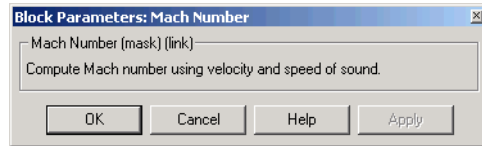
Inputs and Outputs

The first input is the velocity vector.

The second input is the speed of sound.

The output of the block is 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

Purpose

Convert from mass units to desired mass units

Library

Utilities/Unit Conversions

Description

The Mass Conversion block computes the conversion factor from specified input mass units to specified output mass units and applies the conversion factor to the input signal.
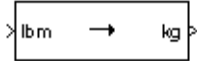
The Mass Conversion block icon displays the input and output units selected from the **Initial units** and the **Final units** lists.

Dialog Box
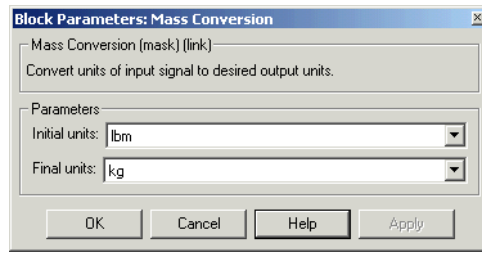
Initial units
Specifies the input units.

Final units
Specifies the output units.

The following conversion units are available:

lbm	Pound mass
kg	Kilograms
slug	Slugs

Inputs and Outputs

The input is the mass in initial mass units.

The output is the mass in final mass units.

See Also

Acceleration Conversion

Angle Conversion

Mass Conversion

Angular Acceleration Conversion

Angular Velocity Conversion

Density Conversion

Force Conversion

Length Conversion

Pressure Conversion

Temperature Conversion

Velocity Conversion

Purpose   Compute moments about center of gravity due to forces that are applied at point CP, not the 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 the center of gravity.

Dialog Box
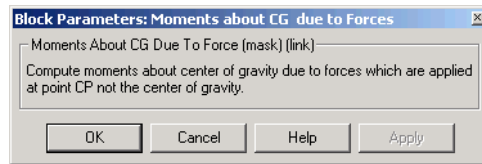
Inputs and Outputs

The first input is the forces applied at point CP.

The second input is the center of gravity.

The third input is the application point of forces.

The output of the block is moments at the center of gravity in x-axes, y-axes and z-axes.

See Also   Aerodynamic Forces and Moments

Estimate Center of Gravity

Non-Standard Day 210C

Purpose   Implement the MIL-STD-210C climatic data

Library   Environment/Atmosphere
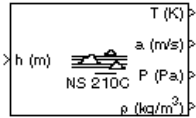
Description   The Non-Standard Day 210C block implements a portion of the climatic data of the MIL-STD-210C worldwide air environment to 80 km (geometric or approximately 262000 feet geometric) for absolute temperature, pressure, density, and speed of sound for the input geopotential altitude.

The Non-Standard Day 210C block icon displays the input and output units selected from the **Units** list.

Dialog Box

Units

Specifies the input and output units:

	Height	**Temperature**	**Speed of Sound**	**Air Pressure**	**Air Density**
`Metric (MKS)`	Meters	Degrees Kelvin	Meters per second	Pascal	Kilograms per cubic meter
`English (Velocity in ft/s)`	Feet	Degrees Rankine	Feet per second	Pound force per square inch	Slug per cubic foot
`English (Velocity in kts)`	Feet	Degrees Rankine	Knots	Pound force per square inch	Slug per cubic foot

Specification

Specify the atmosphere model type from one of the following atmosphere models. The default is `MIL-STD-210C`.

`1976 COESA-extended U.S. Standard Atmosphere`
This selection is linked to the COESA Atmosphere Model block. See the block reference for more information.

`MIL-HDBK-310`
This selection is linked to the Non-Standard Day 310 block. See the block reference for more information.

`MIL-STD-210C`

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

Low temperature

High density

Low density

| 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%	
5%	This option is available only when Envelope is selected for **Atmospheric model type**.
10%	
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

The input is geopotential height.

The four outputs are temperature, speed of sound, air pressure, and air density.

Assumptions and Limitations

All values are held below the geometric altitude of 0 m (0 feet) and above the geometric altitude of 80000 meters (approximately 262000 feet). The envelope atmospheric model has a few exceptions where values are held below the geometric altitude of 1 kilometer (approximately 3281 feet) and above the geometric altitude of 30000 meters (approximately 98425 feet). These exceptions are due to lack of data in MIL-STD-210C for these conditions.

In general, temperature values are extrapolated linearly and density values are extrapolated logarithmically. Pressure and speed of sound are calculated using a perfect gas relationship. The envelope atmospheric model has a few exceptions where the extreme value is linearly interpolated and it is the only value provided as an output. 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 are due to lack of data in MIL-STD-210C for these conditions.

A limitation is that climatic data for the region south of 60°S latitude is excluded from consideration in MIL-STD-210C.

References

Global Climatic Data for Developing Military Products (MIL-STD-210), 9 January 1987, Department of Defense, Washington, D.C.

See Also

COESA Atmosphere Model

ISA Atmosphere Model

Non-Standard Day 310

Non-Standard Day 310

Purpose   Implement the MIL-HDBK-310 climatic data

Library   Environment/Atmosphere

Description
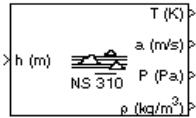
The Non-Standard Day 310 block implements a portion of the climatic data of the MIL-HDBK-310 worldwide air environment to 80 km (geometric or approximately 262000 feet geometric) for absolute temperature, pressure, density, and speed of sound for the input geopotential altitude.

The Non-Standard Day 310 block icon displays the input and output units selected from the **Units** list.

Dialog Box

Units

Specifies the input and output units:

	Height	**Temperature**	**Speed of Sound**	**Air Pressure**	**Air Density**
Metric (MKS)	Meters	Degrees Kelvin	Meters per second	Pascal	Kilograms per cubic meter
English (Velocity in ft/s)	Feet	Degrees Rankine	Feet per second	Pound force per square inch	Slug per cubic foot
English (Velocity in kts)	Feet	Degrees Rankine	Knots	Pound force per square inch	Slug per cubic foot

Specification

Specify the atmosphere model type from one of the following atmosphere models. The default is MIL-HDBK-310.

1976 COESA-extended U.S. Standard Atmosphere
This selection is linked to the COESA Atmosphere Model block. See the block reference for more information.

MIL-HDBK-310

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

Low temperature

High density

Low density

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%	
5%	This option is available only when Envelope is selected for **Atmospheric model type**.
10%	
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

The input is geopotential height.

The four outputs are temperature, speed of sound, air pressure, and air density.

Assumptions and Limitations

All values are held below the geometric altitude of 0 m (0 feet) and above the geometric altitude of 80000 meters (approximately 262000 feet). The envelope atmospheric model has a few exceptions where values are held below the geometric altitude of 1 kilometer (approximately 3281 feet) and above the geometric altitude of 30000 meters (approximately 98425 feet). These exceptions are due to lack of data in MIL-HDBK-310 for these conditions.

In general, temperature values are extrapolated linearly and density values are extrapolated logarithmically. Pressure and speed of sound are calculated using a perfect gas relationship. The envelope atmospheric model has a few exceptions where the extreme value is linearly interpolated and it is the only value provided as an output. 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 are due to lack of data in MIL-HDBK-310 for these conditions.

A limitation is that climatic data for the region south of 60°S latitude is excluded from consideration in MIL-HDBK-310.

References

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

Pressure Altitude

Purpose
Calculate pressure altitude based on ambient pressure

Library
Environment/Atmosphere

Description
The Pressure Altitude block computes the pressure altitude based on ambient pressure. Pressure altitude is the altitude in the 1976 Committee on the Extension of the Standard Atmosphere (COESA) United States with specified ambient pressure.
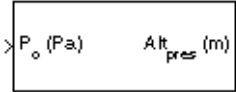
Pressure altitude is also known as the mean sea level (MSL) altitude.

The Pressure Altitude block icon displays the input and output units selected from the **Units** list.

Dialog Box
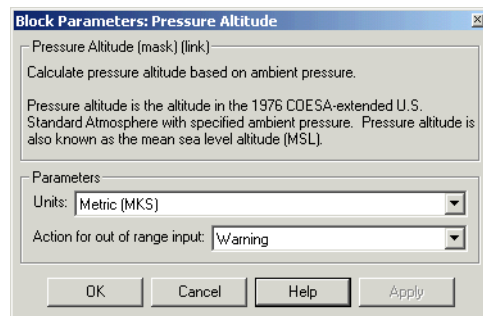
Units
Specifies the input units:

	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
The input is the static pressure.

The output is 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.

References　　U.S. Standard Atmosphere, 1976, U.S. Government Printing Office, Washington, D.C.

See Also　　COESA Atmosphere Model

Pressure Conversion

Purpose   Convert from pressure units to desired pressure units

Library   Utilities/Unit Conversions

Description   The Pressure Conversion block computes the conversion factor from specified input pressure units to specified output pressure units and applies the conversion factor to the input signal.
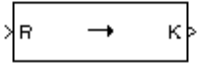
The Pressure Conversion block icon displays the input and output units selected from the **Initial units** and the **Final units** lists.

Dialog Box
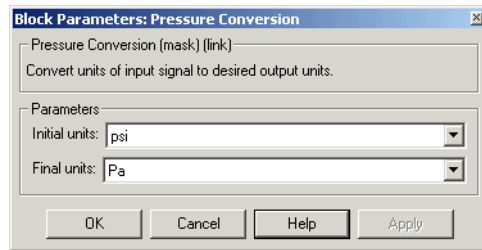
Initial units
   Specifies the input units.

Final units
   Specifies the output units.

The following conversion units are available:

psi	Pound mass per square inch
Pa	Pascals
psf	Pound mass per square foot
atm	Atmospheres

Inputs and Outputs   The input is the pressure in initial pressure units.

The output is 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

Quaternions to Direction Cosine Matrix

Purpose   Convert quaternion vector to direction cosine matrix

Library   Utilities/Axes Transformations
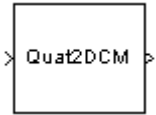
Description   The Quaternions to Direction Cosine Matrix block transforms the four-element
unit quaternion vector (q_0, q_1, q_2, q_3) into a 3-by-3 direction cosine matrix
(DCM). The outputted DCM performs the coordinate transformation of a vector
in inertial axes to a vector in body axes.

Using quaternion algebra, if a point P is subject to the rotation described by a
quaternion q, it changes to P' given by the following relationship:

$$P' = qPq^c$$
$$q = q_0 + \boldsymbol{i}q_1 + \boldsymbol{j}q_2 + \boldsymbol{k}q_3$$
$$q^c = q_0 - \boldsymbol{i}q_1 - \boldsymbol{j}q_2 - \boldsymbol{k}q_3$$
$$P = 0 + \boldsymbol{i}x + \boldsymbol{j}y + \boldsymbol{k}z$$

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.

$$DCM = \begin{bmatrix} (q_0^2 + q_1^2 - q_2^2 - q_3^2) & 2(q_1 q_2 + q_0 q_3) & 2(q_1 q_3 - q_0 q_2) \\ 2(q_1 q_2 - q_0 q_3) & (q_0^2 - q_1^2 + q_2^2 - q_3^2) & 2(q_2 q_3 + q_0 q_1) \\ 2(q_1 q_3 + q_0 q_2) & 2(q_2 q_3 - q_0 q_1) & (q_0^2 - q_1^2 - q_2^2 + q_3^2) \end{bmatrix}$$

Dialog Box
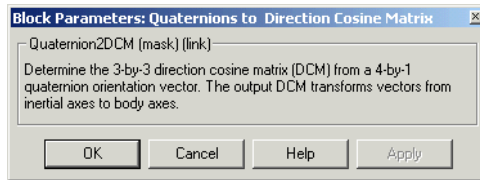
Block Parameters: Quaternions to Direction Cosine Matrix

Quaternion2DCM (mask) (link)

Determine the 3-by-3 direction cosine matrix (DCM) from a 4-by-1 quaternion orientation vector. The output DCM transforms vectors from inertial axes to body axes.

| OK | Cancel | Help | Apply |

Inputs and Outputs

The input is a 4-by-1 quaternion vector.

The output is a 3-by-3 direction cosine matrix.

See Also

Direction Cosine Matrix to Euler Angles

Direction Cosine Matrix to Quaternions

Euler Angles to Direction Cosine Matrix

Euler Angles to Quaternions

Quaternions to Euler Angles

Quaternions to Euler Angles

Purpose   Convert quaternion vector to Euler angles

Library   Utilities/Axes Transformations
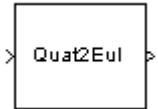
Description   The Quaternions to Euler Angles block converts the four-element unit
quaternion (q_0,q_1,q_2,q_3) into the equivalent three Euler angle rotations (roll,
pitch, yaw).

The conversion is generated by comparing elements in the direction cosine
matrix (DCM), as functions of the Euler rotation angles, with elements in the
DCM, as functions of a unit quaternion vector:

$$DCM = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ (\sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi) & (\sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi) & \sin\phi\cos\theta \\ (\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi) & (\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi) & \cos\phi\cos\theta \end{bmatrix}$$

$$DCM = \begin{bmatrix} (q_0^2 + q_1^2 - q_2^2 - q_3^2) & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & (q_0^2 - q_1^2 + q_2^2 - q_3^2) & 2(q_2q_3 + q_0q_1) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & (q_0^2 - q_1^2 - q_2^2 + q_3^2) \end{bmatrix}$$

From the preceding, you can derive the following relationships between DCM
elements and individual Euler angles:

$$\begin{aligned} \phi &= \operatorname{atan}(DCM(2,3), DCM(3,3)) \\ &= \operatorname{atan}(2(q_2q_3 + q_0q_1), (q_0^2 - q_1^2 - q_2^2 + q_3^2)) \\ \theta &= \operatorname{asin}(-DCM(1,3)) \\ &= \operatorname{asin}(-2(q_1q_3 - q_0q_2)) \\ \psi &= \operatorname{atan}(DCM(1,2), DCM(1,1)) \\ &= \operatorname{atan}(2(q_1q_2 + q_0q_3), (q_0^2 + q_1^2 - q_2^2 - q_3^2)) \end{aligned}$$

Dialog Box
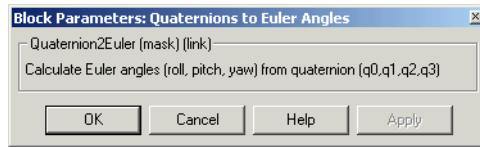
Inputs and Outputs

The input is a 4-by-1 quaternion vector.

The output is a 3-by-1 vector of Euler angles.

Assumptions and Limitations

This implementation generates a pitch angle that lies between ±90 degrees, and roll and yaw angles that lie between ±180 degrees.

The Euler angle solution is singular when the pitch angle θ is equal to ±90 degrees.

Examples

See aero_six_dof for an example of the use of the Quaternions to Euler Angles block in an implementation of the equations of motion of a rigid body.

See Also

Direction Cosine Matrix to Euler Angles

Direction Cosine Matrix to Quaternions

Euler Angles to Direction Cosine Matrix

Euler Angles to Quaternions

Quaternions to Direction Cosine Matrix

Relative Ratio

Purpose

Calculate relative atmospheric ratios

Library

Flight Parameters

Description

The Relative Ratio block computes the relative atmospheric ratios, including relative temperature ratio (θ), $\sqrt{\theta}$, relative pressure ratio (δ), and relative density ratio (σ).

θ represents the ratio of the air stream temperature at a chosen reference station relative to sea level standard atmospheric conditions.

$$\theta = \frac{T}{T_o}$$

δ represents the ratio of the air stream pressure at a chosen reference station relative to sea level standard atmospheric conditions.

$$\delta = \frac{P}{P_o}$$

σ represents the ratio of the air stream density at a chosen reference station relative to sea level standard atmospheric conditions.

$$\sigma = \frac{\rho}{\rho_o}$$

The Relative Ratio block icon displays the input units selected from the **Units** list.

Dialog Box
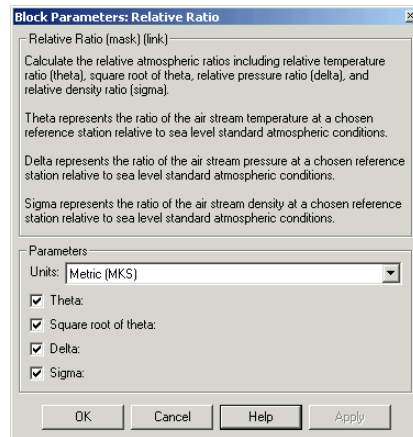
Units

Specifies the input units:

	Tstatic	**Pstatic**	**rho_static**
Metric (MKS)	Degrees Kelvin	Pascal	Kilograms per cubic meter
English	Degrees Rankine	Pound force per square inch	Slug per cubic foot

Theta

When selected, the θ is calculated and static temperature is a required input.

Square root of theta

When selected, the $\sqrt{\theta}$ is calculated and static temperature is a required input.

Delta

When selected, the δ is calculated and static pressure is a required input.

Sigma

When selected, the σ is calculated and static density is a required input.

Inputs and Outputs

The four possible inputs are Mach number, static temperature, static pressure, and static density.

The four possible outputs are θ, $\sqrt{\theta}$, δ, and σ.

Assumptions

For cases in which total temperature, total pressure, or total density ratio is desired (Mach number is nonzero), the total temperature, total pressure, and total densities are calculated assuming perfect gas (with constant molecular weight, constant pressure specific heat, and constant specific heat ratio) and dry air.

References

Aeronautical Vestpocket Handbook, United Technologies Pratt & Whitney, August, 1986.

Second Order Linear Actuator

Purpose   Implement a 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 parameters that
define the system.

Dialog Box
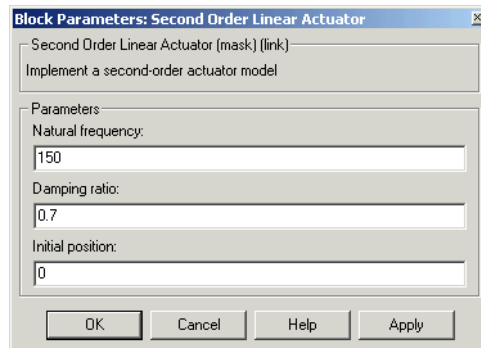
Natural frequency

The natural frequency of the actuator. The units of natural frequency are
radians per second.

Damping ratio

The damping ratio of the actuator. A dimensionless parameter.

Initial position

The initial position of the actuator. The units of initial position should be
the same as the units of demanded actuator position.

**Inputs and
Outputs**   The input is the demanded actuator position.

The output is the actual actuator position.

See Also   Second Order Nonlinear Actuator

Purpose   Implement a 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
parameters that define the system.

Dialog Box

Natural frequency

The natural frequency of the actuator. The units of natural frequency are
radians per second.

Damping ratio

The damping ratio of the actuator. A dimensionless parameter.

Maximum deflection

The largest actuator position allowable. The units of maximum deflection
should be the same as the units of demanded actuator position.

Second Order Nonlinear Actuator

Minimum deflection

The smallest actuator position allowable. The units of minimum deflection should be the same as the units of demanded actuator position.

Maximum rate

The fastest speed allowable for actuator motion. The units of maximum rate should be the units of demanded actuator position per second.

Initial position

The initial position of the actuator. The units of initial position should be the same as the units of demanded actuator position.

Inputs and Outputs

The input is the demanded actuator position.

The output is the actual actuator position.

Examples

See the aero_guidance model and the Actuators subsystem in the aeroblk_HL20 model for an example of this block.

See Also

Second Order Linear Actuator

Purpose   Implement a state-space controller in a self-conditioned form

Library   GNC/Controls

Description   The Self-Conditioned [A,B,C,D] block can be used to implement the state-space controller defined by
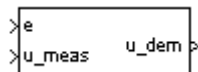
$$\begin{bmatrix} \dot{x} = Ax + Be \\ u = Cx + De \end{bmatrix}$$

in the self-conditioned form

$$\dot{z} = (A - HC)z + (B - HD)e + Hu_{meas}$$

$$u_{dem} = Cz + De$$

The input u_{meas} is a vector of the achieved actuator positions, and the output u_{dem} is the vector of controller actuator demands. In the case that the actuators are not limited, then $u_{meas} = udem$ and substituting the output equation into the state equation returns the nominal controller. In the case that they are not equal, the dynamics of the controller are set by the poles of A-HC.

Hence H must be chosen to make the poles sufficiently fast to track u_{meas} but at the same time not so fast that noise on e is propagated to u_{dem}. The matrix H is designed by a callback to the Control System Toolbox command `place` to place the poles at defined locations.

Self-Conditioned [A,B,C,D]

Dialog Box
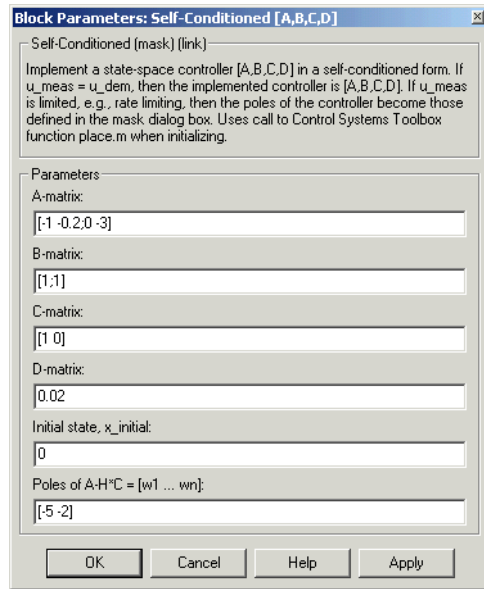
A-matrix

A-matrix of the state-space implementation.

B-matrix

B-matrix of the state-space implementation.

C-matrix

C-matrix of the state-space implementation.

D-matrix

D-matrix of the state-space implementation.

Initial state, x_initial

This is a vector of initial states for the controller, i.e., initial values for the state vector, z. It should have length equal to the size of the first dimension of A.

Poles of A-H*C

This is a vector of the desired poles of A-H*C. Hence the number of pole locations defined should be equal to the dimension of the A-matrix.

Inputs and Outputs

The first input is the control error.

The second input is the measured actuator position.

The output is the actuator demands.

Assumptions and Limitations

This block requires the Control System Toolbox.

Examples

This Simulink model shows a state-space controller implemented in both self-conditioned and standard state-space forms. The actuator authority limits of +/- 0.5 units are modeled by the saturation block.
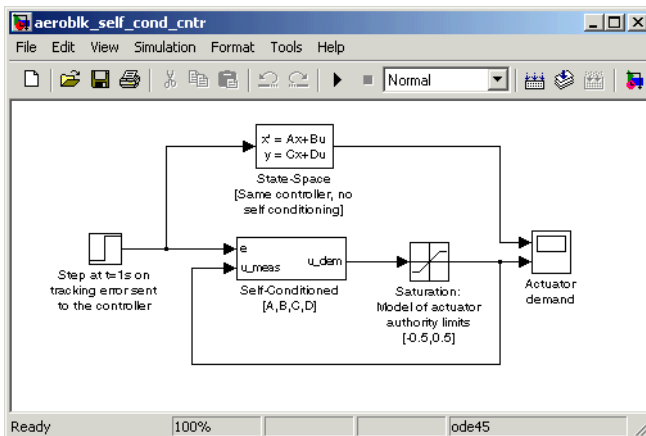
Self-Conditioned [A,B,C,D]

Notice that the *A*-matrix has a zero in the 1,1 element, indicating integral action.
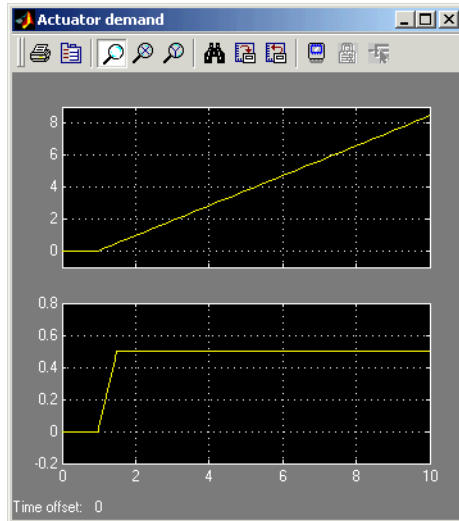
The top trace shows the conventional state-space implementation. The output of the controller winds up well past the actuator upper authority limit of +0.5. The lower trace shows that the self-conditioned form results in an actuator demand that tracks the upper authority limit, which means that when the sign of the control error, e, is reversed, the actuator demand responds immediately.

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

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

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

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

Purpose   Implement three-degrees-of-freedom equations of motion

Library   Equations of Motion/3DoF

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

Simple Variable Mass 3DoF (Body Axes)

The equations of motion are

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

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

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

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

$$\dot{I}_{yy} = \frac{I_{yyfull} - I_{yy\,empty}}{m_{full} - m_{empty}}\dot{m}$$

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

Dialog Box

Block Parameters: Simple Variable Mass 3DoF (Body Axes)

3DoF EoM (mask) (link)

Integrate the three-degrees-of-freedom equations of motion to determine body position, velocity, attitude, and related values.

Parameters

Units: Metric (MKS)

Mass type: Simple Variable

Initial velocity:

100

Initial body attitude:

0

Initial incidence:

0

Initial body rotation rate:

0

Initial position (x z):

[0 0]

Initial mass:

1.0

Empty mass:

0.5

Full mass:

3.0

Empty inertia:

0.5

Full inertia:

3.0

Gravity source: External

OK   Cancel   Help   Apply

Units

Specifies the input and output units:

	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:

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

The Simple Variable selection conforms to the previously described equations of motion.

Initial velocity

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

Initial body attitude

A scalar value for the initial pitch attitude of the body, (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 Earth-fixed reference frame.

Initial mass

A scalar value for the initial mass of the body.

Empty mass

A scalar value for the empty mass of the body.

Full mass

A scalar value for the full mass of the body.

Empty inertia

A scalar value for the empty inertia of the body.

Full inertia

A scalar value for the full inertia of the body.

Gravity source

Specify source of gravity:

External	Variable gravity input to block
Internal	Constant gravity specified in mask

Acceleration due to gravity

A scalar value for the acceleration due to gravity used if internal gravity source is selected. If gravity is to be neglected in the simulation, this value can be set to 0.

Inputs and Outputs

The first input to the block is the force acting along the body x-axis, (F_x).

The second input to the block is the force acting along the body z-axis, (F_z).

The third input to the block is the applied pitch moment, (M).

The fourth input to the block is the rate of change of mass, $(\dot{m})$.

The fifth optional input to the block is gravity in the selected units.

The first output from the block is the pitch attitude, in radians (θ).

Simple Variable Mass 3DoF (Body Axes)

The second output is the pitch angular rate, in radians per second (q).

The third output is the pitch angular acceleration, in radians per second squared ($\dot{q}$).

The fourth output is a two-element vector containing the location of the body, in the Earth-fixed reference frame, (*Xe,Ze*).

The fifth output is a two-element vector containing the velocity of the body resolved into the body-fixed coordinate frame, (*u,w*).

The sixth output is a two-element vector containing the acceleration of the body resolved into the body-fixed coordinate frame, (*Ax,Az*).

The seventh output is a scalar element containing a flag for fuel tank status, (*Fuel*):

- 1 indicates that the tank is full.
- 0 indicates that the integral is neither full nor empty.
- -1 indicates that the tank is empty.

See Also

3DoF (Body Axes)

Custom Variable Mass 3DoF (Body Axes)

Incidence & Airspeed

Simple Variable Mass 6DoF (Euler Angles)

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

Library   Equations of Motion/6DoF

Description   The Simple Variable Mass 6DoF (Euler Angles) block considers the rotation of a body-fixed coordinate frame (X_b, Y_b, Z_b) about an Earth-fixed 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 Earth-fixed reference frame is considered inertial, a simplification that allows the forces due to the Earth's motion relative to a star-fixed reference system to be neglected.

Earth-fixed 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.

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

Simple Variable Mass 6DoF (Euler Angles)

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

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

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

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

The inertia tensor is determined using a table lookup which linearly interpolates between I_{full} and I_{empty} based on mass (m). While the rate of change of the inertia tensor is estimated by the following equation.

$$\dot{I} = \frac{I_{full} - I_{empty}}{m_{full} - m_{empty}} \dot{m}$$

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

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

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

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

Dialog Box

Block Parameters: Simple Variable Mass 6DoF (Euler Angles)

6DoF EoM (Body Axis) (mask) (link)

Integrate the six-degrees-of-freedom equations of motion using an Euler angle representation for the orientation of the body in space.

Parameters

Units: Metric (MKS)

Mass type: Simple Variable

Representation: Euler Angles

Initial position in inertial axes [Xe,Ye,Ze]:

[0 0 0]

Initial velocity in body axes [U,v,w]:

[0 0 0]

Initial Euler orientation [roll, pitch, yaw]:

[0 0 0]

Initial body rotation rates [p,q,r]:

[0 0 0]

Initial mass:

1.0

Empty mass:

0.5

Full mass:

2.0

Empty inertia matrix:

eye(3)

Full inertia matrix:

2*eye(3)

OK   Cancel   Help   Apply

Simple Variable Mass 6DoF (Euler Angles)

Units

Specifies the input and output 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:

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

The Simple Variable selection conforms to the previously described equations of motion.

Representation

Select the representation to use:

Mass	Description
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 Earth-fixed reference frame.

Initial velocity in body axes

The three-element vector for the initial velocity in the body-fixed coordinate frame.

Initial Euler rotation

The three-element vector for the initial Euler rotation angles [roll, pitch, yaw], in radians.

Initial body rotation rates

The three-element vector for the initial body-fixed angular rates, in radians per second.

Initial mass

The initial mass of the rigid body.

Empty mass

A scalar value for the empty mass of the body.

Full mass

A scalar value for the full mass of the body.

Empty inertia matrix

A 3-by-3 inertia tensor matrix for the empty inertia of the body.

Full inertia matrix

A 3-by-3 inertia tensor matrix for the full inertia of the body.

Inputs and Outputs

The first input to the block is a vector containing the three applied forces.

The second input is a vector containing the three applied moments.

The third input is a scalar containing the rate of change of mass.

The first output is a three-element vector containing the velocity in the Earth-fixed reference frame.

The second output is a three-element vector containing the position in the Earth-fixed reference frame.

Simple Variable Mass 6DoF (Euler Angles)

The third output is a three-element vector containing the Euler rotation angles [roll, pitch, yaw], in radians.

The fourth output is a 3-by-3 matrix for the coordinate transformation from Earth-fixed axes to body-fixed axes.

The fifth output is a three-element vector containing the velocity in the body-fixed frame.

The sixth output is a three-element vector containing the angular rates in body-fixed axes, in radians per second.

The seventh output is a three-element vector containing the angular accelerations in body-fixed axes, in radians per second.

The eighth output is a three-element vector containing the accelerations in body-fixed axes.

The ninth output is a scalar element containing a flag for fuel tank status:

- 1 indicates that the tank is full.
- 0 indicates that the integral is neither full nor empty.
- -1 indicates that the tank is empty.

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

References Mangiacasale, L., "Flight Mechanics of a u-Airplane with a MATLAB Simulink Helper," Edizioni Libreria CLUP, 1998.

See Also 6DoF (Euler Angles)

6DoF (Quaternion)

Custom Variable Mass 6DoF (Euler Angles)\

Custom Variable Mass 6DoF (Quaternion)

Simple Variable Mass 6DoF (Quaternion)

Purpose

Implement a quaternion 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 Simple Variable Mass 6DoF (Euler Angles) block.

The integration of the rate of change of the quaternion vector is given below. The gain K drives the norm of the quaternion state vector to 1.0 should 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.

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

$$\varepsilon = 1 - (q_0^2 + q_1^2 + q_3^2 + q_4^2)$$

Simple Variable Mass 6DoF (Quaternion)

Dialog Box

Block Parameters: Simple Variable Mass 6DoF (Quaternion)

6DoF EoM (Body Axis) (mask) (link)

Integrate the six-degrees-of-freedom equations of motion using an Euler angle representation for the orientation of the body in space.

Parameters

Units: Metric (MKS)

Mass type: Simple Variable

Representation: Quaternion

Initial position in inertial axes [Xe,Ye,Ze]:

[0 0 0]

Initial velocity in body axes [U,v,w]:

[0 0 0]

Initial Euler orientation [roll, pitch, yaw]:

[0 0 0]

Initial body rotation rates [p,q,r]:

[0 0 0]

Initial mass:

1.0

Empty mass:

0.5

Full mass:

2.0

Empty inertia matrix:

eye(3)

Full inertia matrix:

2*eye(3)

Gain for quaternion normalization:

1.0

| OK | Cancel | Help | Apply |

Units

Specifies the input and output 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:

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

The Simple Variable selection conforms to the previously described equations of motion.

Representation

Select the representation to use:

Mass	Description
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.

Simple Variable Mass 6DoF (Quaternion)

Initial position in inertial axes

The three-element vector for the initial location of the body in the Earth-fixed reference frame.

Initial velocity in body axes

The three-element vector for the initial velocity in the body-fixed coordinate frame.

Initial Euler rotation

The three-element vector for the initial Euler rotation angles [roll, pitch, yaw], in radians.

Initial body rotation rates

The three-element vector for the initial body-fixed angular rates, in radians per second.

Initial mass

The initial mass of the rigid body.

Empty mass

A scalar value for the empty mass of the body.

Full mass

A scalar value for the full mass of the body.

Empty inertia matrix

A 3-by-3 inertia tensor matrix for the empty inertia of the body.

Full inertia matrix

A 3-by-3 inertia tensor matrix for the full inertia of the body.

Gain for quaternion normalization

The gain to maintain the norm of the quaternion vector equal to 1.0.

Inputs and Outputs

The first input to the block is a vector containing the three applied forces.

The second input is a vector containing the three applied moments.

The third input is a scalar containing the rate of change of mass.

The first output is a three-element vector containing the velocity in the Earth-fixed reference frame.

The second output is a three-element vector containing the position in the Earth-fixed reference frame.

The third output is a three-element vector containing the Euler rotation angles [roll, pitch, yaw], in radians.

The fourth output is a 3-by-3 matrix for the coordinate transformation from Earth-fixed axes to body-fixed axes.

The fifth output is a three-element vector containing the velocity in the body-fixed frame.

The sixth output is a three-element vector containing the angular rates in body-fixed axes, in radians per second.

The seventh output is a three-element vector containing the angular accelerations in body-fixed axes, in radians per second.

The eighth output is a three-element vector containing the accelerations in body-fixed axes.

The ninth output is a scalar element containing a flag for fuel tank status:

- 1 indicates that the tank is full.
- 0 indicates that the integral is neither full nor empty.
- -1 indicates that the tank is empty.

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

References   Mangiacasale, L., "Flight Mechanics of a u-Airplane with a MATLAB Simulink Helper," Edizioni Libreria CLUP, 1998.

See Also   6DoF (Euler Angles)

6DoF (Quaternion)

Custom Variable Mass 6DoF (Euler Angles)

Custom Variable Mass 6DoF (Quaternion)

Simple Variable Mass 6DoF (Euler Angles)

SinCos

Purpose   Compute the sine and cosine of the input angle

Library   Utilities/Math Operations

Description   The SinCos block computes the sine and cosine of the input angle, theta.

```
sin(u)
cos(u)
```

Dialog Box

```
Block Parameters: SinCos                            ×
─ SinCos (mask) (link)──────────────────────────────
  Compute the sine and cosine of input, u. u is in radians.

     [    OK    ]   [  Cancel  ]   [   Help   ]   [  Apply  ]
```
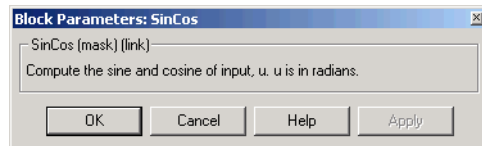
Inputs and Outputs

The first input is an angle, in radians.

The first output is the sine of the input angle.

The second output is the cosine of the input angle.

Purpose   Create an inertia tensor from moments and products of inertia

Library   Mass Properties
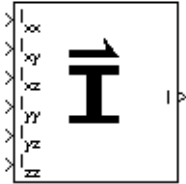
Description   The Symmetric Inertia Tensor block creates an inertia tensor from moments and products of inertia. Each input corresponds to an element of the tensor.

The inertia tensor has the form of

$$Inertia = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{yz} \\ -I_{xy} & I_{yy} & -I_{xz} \\ -I_{yz} & -I_{xz} & I_{zz} \end{bmatrix}$$

Dialog Box

Block Parameters: Symmetric Inertia Tensor

Symmetric Inertia Tensor (mask) (link)

Create an inertia tensor from moments and products of inertia. Each input corresponds to an element of the tensor.

    OK   Cancel   Help   Apply
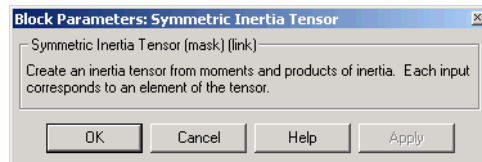
Inputs and Outputs   The first input is the moment of inertia about the *x-axis*.

The second input is the product of inertia in the xy plane.

The third input is the product of inertia in the xz plane.

The fourth input is the moment of inertia about the *y-axis*.

The fifth input is the product of inertia in the yz plane.

The sixth input is the moment of inertia about the *z-axis*.

The output of the block is a symmetric 3-by-3 inertia tensor.

See Also   Create 3x3 Matrix

Temperature Conversion

Purpose   Convert from temperature units to desired temperature units

Library   Utilities/Unit Conversions
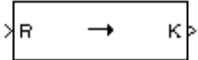
Description   The Temperature Conversion block computes the conversion factor from specified input temperature units to specified output temperature units and applies the conversion factor to the input signal.

The Temperature Conversion block icon displays the input and output units selected from the **Initial units** and the **Final units** lists.

Dialog Box
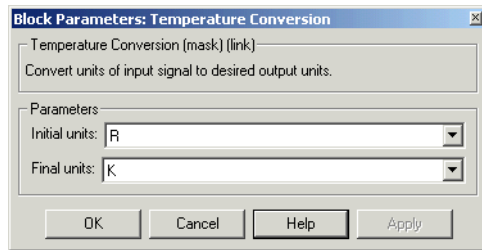
Initial units
Specifies the input units.

Final units
Specifies the output units.

The following conversion units are available:

K	Degrees Kelvin
F	Degrees Fahrenheit
C	Degrees Celsius
R	Degrees Rankine

Inputs and Outputs   The input is the temperature in initial temperature units.

The output is the temperature in final temperature units.

See Also   Acceleration Conversion

Angle Conversion

Angular Acceleration Conversion

Angular Velocity Conversion

Density Conversion

Force Conversion

Length Conversion

Mass Conversion

Pressure Conversion

Velocity Conversion

Turbofan Engine System

Purpose   Implement a first-order representation of a turbofan engine with controller

Library   Propulsion
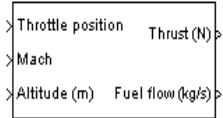
Description   The Turbofan Engine System block computes the thrust and the weight of fuel
flow of a turbofan engine and controller at a specific throttle position, Mach
number, and altitude.

This system is represented by a first-order system with unitless heuristic
lookup tables for thrust, thrust specific fuel consumption (TSFC), and engine
time constant. For the lookup table data, thrust is a function of throttle position
and Mach number, TSFC is a function of thrust and Mach number, and engine
time constant is a function of thrust. The unitless lookup table outputs are
corrected for altitude using the relative pressure ratio δ and relative
temperature ratio θ, and scaled by maximum sea level static thrust, fastest
engine time constant at sea level static, sea level static thrust specific fuel
consumption, and ratio of installed thrust to uninstalled thrust.

The Turbofan Engine System block icon displays the input and output units
selected from the **Units** list.

Dialog Box

Units

Specifies the input and output units:

	Altitude	**Thrust**	**Fuel Flow**
Metric (MKS)	Meters	Newtons	Kilograms per second
English	Feet	Pound force	Pound mass per second

Initial thrust source

Specifies the source of initial thrust:

Internal	Use initial thrust value from mask dialog.
External	Use external input for initial thrust value.

Turbofan Engine System

Initial thrust

Initial value for thrust.

Maximum sea-level static thrust

Maximum thrust at sea-level and at Mach = 0.

Fastest engine time constant at sea-level static

Fastest engine time at sea level.

Sea-level static thrust specific fuel consumption

Thrust specific fuel consumption at sea level, at Mach = 0, and at maximum thrust, in specified mass units per hour per specified thrust units.

Ratio of installed thrust to uninstalled thrust

Coefficient representing the loss in thrust due to engine installation.

Inputs and Outputs

The first input is the throttle position. Throttle position can vary from zero to one, corresponding to no to full throttle.

The second input is the Mach number.

The third input is the altitude in specified length units.

The first output is the thrust in specified force units.

The second output is the fuel flow in specified mass units per second.

Assumptions and Limitations

The atmosphere is at standard day conditions and an ideal gas.

The Mach number is limited to less than 1.0.

This engine system is for indication purposes only. It is not meant to be used as a reference model.

References

"Aeronautical Vestpocket Handbook," United Technologies Pratt & Whitney, August, 1986.

Raymer, D. P., "Aircraft Design: A Conceptual Approach," AIAA Education Series, Washington, DC, 1989.

Hill, P. G., and C. R. Peterson, *Mechanics and Thermodynamics of Propulsion,* Addison-Wesley Publishing Company, Reading, MA, 1970.

Purpose   Convert from velocity units to desired velocity units

Library   Utilities/Unit Conversions

Description   The Velocity Conversion block computes the conversion factor from specified
input velocity units to specified output velocity units and applies the
conversion factor to the input signal.

The Velocity Conversion block icon displays the input and output units selected
from the **Initial units** and the **Final units** lists.

Dialog Box
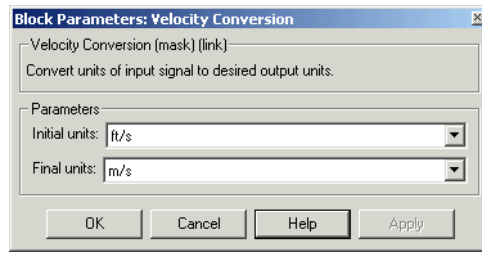
Initial units
   Specifies the input units.

Final units
   Specifies the output units.

The following conversion units are available:

m/s	Meters per second
ft/s	Feet per second
km/s	Kilometers per second
in/s	Inches per second
km/h	Kilometers per hour
mph	Miles per hour
kts	Nautical miles per hour

Velocity Conversion

Inputs and Outputs

The input is the velocity in initial velocity units.

The output is the velocity in final velocity units.

See Also

Acceleration Conversion

Angle Conversion

Angular Acceleration Conversion

Angular Velocity Conversion

Density Conversion

Force Conversion

Length Conversion

Mass Conversion

Pressure Conversion

Temperature Conversion

Von Karman Wind Turbulence Model (Continuous)

Purpose   Generate continuous wind turbulence with the Von Kármán velocity spectra

Library   Environment/Wind
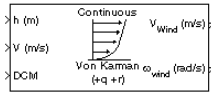
Description   The Von Kármán Wind Turbulence Model (Continuous) block uses the Von Kármán spectral representation to add turbulence to the aerospace model by passing band-limited white noise through appropriate forming filters. This block implements the mathematical representation in the Military Specification MIL-F-8785C and Military Handbook MIL-HDBK-1797.

According to the military references, turbulence is a stochastic process defined by velocity spectra. For an aircraft flying at a speed V through a "frozen" turbulence field with a spatial frequency of 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**
Longitudinal		
$\Phi_u(\omega)$	$\dfrac{2\sigma_u^2 L_u}{\pi V} \cdot \dfrac{1}{[1 + (1.339 L_u \frac{\omega}{V})^2]^{5/6}}$	$\dfrac{2\sigma_u^2 L_u}{\pi V} \cdot \dfrac{1}{[1 + (1.339 L_u \frac{\omega}{V})^2]^{5/6}}$
$\Phi_p(\omega)$	$\dfrac{\sigma_w^2}{VL_w} \cdot \dfrac{0.8 \left(\frac{\pi L_w}{4b}\right)^{\frac{1}{3}}}{1 + \left(\frac{4b\omega}{\pi V}\right)^2}$	$\dfrac{\sigma_w^2}{2VL_w} \cdot \dfrac{0.8 \left(\frac{2\pi L_w}{4b}\right)^{\frac{1}{3}}}{1 + \left(\frac{4bw}{\pi V}\right)^2}$

Von Karman Wind Turbulence Model (Continuous)

	MIL-F-8785C	**MIL-HDBK-1797**
Lateral		
$\Phi_v(\omega)$	$\dfrac{\sigma_v^2 L_v}{\pi V} \cdot \dfrac{1 + \frac{8}{3}(1.339 L_v \frac{\omega}{V})^2}{[1 + (1.339 L_v \frac{\omega}{V})^2]^{11/6}}$	$\dfrac{2\sigma_v^2 L_v}{\pi V} \cdot \dfrac{1 + \frac{8}{3}(2.678 L_v \frac{\omega}{V})^2}{[1 + (2.678 L_v \frac{\omega}{V})^2]^{11/6}}$
$\Phi_r(\omega)$	$\dfrac{\mp \left(\frac{\omega}{V}\right)^2}{1 + \left(\frac{3b\omega}{\pi V}\right)^2} \cdot \Phi_v(\omega)$	$\dfrac{\mp \left(\frac{\omega}{V}\right)^2}{1 + \left(\frac{3b\omega}{\pi V}\right)^2} \cdot \Phi_v(\omega)$
Vertical		
$\Phi_w(\omega)$	$\dfrac{\sigma_w^2 L_w}{\pi V} \cdot \dfrac{1 + \frac{8}{3}(1.339 L_w \frac{\omega}{V})^2}{[1 + (1.339 L_w \frac{\omega}{V})^2]^{11/6}}$	$\dfrac{2\sigma_w^2 L_w}{\pi V} \cdot \dfrac{1 + \frac{8}{3}(2.678 L_w \frac{\omega}{V})^2}{[1 + (2.678 L_w \frac{\omega}{V})^2]^{11/6}}$
$\Phi_q(\omega)$	$\dfrac{\pm \left(\frac{\omega}{V}\right)^2}{1 + \left(\frac{4b\omega}{\pi V}\right)^2} \cdot \Phi_w(\omega)$	$\dfrac{\pm \left(\frac{\omega}{V}\right)^2}{1 + \left(\frac{4b\omega}{\pi V}\right)^2} \cdot \Phi_w(\omega)$

The variable b represents the aircraft wingspan. The variables L_u, L_v, L_w represent the turbulence scale lengths. The variables $\sigma_u, \sigma_v, \sigma_w$ represent the turbulence intensities:

The spectral density definitions of turbulence angular rates are defined in the references as three variations, which are displayed in the following table:

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

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

$$q_g = \frac{\partial w_g}{\partial x}$$

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

The variations affect only the vertical (q_g) and lateral (r_q) 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
$\Phi_q(\omega)$	$-\Phi_r(\omega)$
$\Phi_q(\omega)$	$\Phi_r(\omega)$
$-\Phi_q(\omega)$	$\Phi_r(\omega)$

Von Karman Wind Turbulence Model (Continuous)

To generate a signal with the correct characteristics, a unit variance, band-limited white noise signal is passed through forming filters. The forming filters are approximations of the Von Kármán velocity spectra which are valid in a range of normalized frequencies of less than 50 radians. These filters can be found in both the Military Handbook MIL-HDBK-1797 and the reference by Ly and Chan.

The following table displays the transfer functions:

MIL-F-8785C/MIL-HDBK-1797

Longitudinal

$H_u(s)$

$$\frac{\sigma_u \sqrt{\frac{2}{\pi} \cdot \frac{L_u}{V}} \left(1 + 0.25\frac{L_u}{V}s\right)}{1 + 1.357\frac{L_u}{V}s + 0.1987(\frac{L_u}{V})^2 s^2}$$

$H_p(s)$

$$\sigma_w \sqrt{\frac{0.8}{V}} \frac{\left(\frac{\pi}{(4b)}\right)^{1/6}}{L_w^{1/3}\left(1 + \left(\frac{4b}{\pi V}\right)s\right)}$$

Lateral

$H_v(s)$

$$\frac{\sigma_v \sqrt{\frac{1}{\pi} \cdot \frac{L_v}{V}} \left(1 + 2.7478\frac{L_v}{V}s + 0.3398(\frac{L_v}{V})^2 s^2\right)}{1 + 2.9958\frac{L_v}{V}s + 1.9754(\frac{L_v}{V})^2 s^2 + 0.1539(\frac{L_v}{V})^3 s^3}$$

$H_r(s)$

$$\frac{\mp\frac{s}{V}}{\left(1 + \left(\frac{3b}{\pi V}\right)s\right)} \cdot H_v(s)$$

Von Karman Wind Turbulence Model (Continuous)

MIL-F-8785C/MIL-HDBK-1797

Vertical

$H_w(s)$
$$\frac{\sigma_w \sqrt{\frac{1}{\pi} \cdot \frac{L_w}{V}} \left(1 + 2.7478 \frac{L_w}{V} s + 0.3398 \left(\frac{L_w}{V}\right)^2 s^2\right)}{1 + 2.9958 \frac{L_w}{V} s + 1.9754 \left(\frac{L_w}{V}\right)^2 s^2 + 0.1539 \left(\frac{L_w}{V}\right)^3 s^3}$$

$H_q(s)$
$$\frac{\pm \frac{s}{V}}{\left(1 + \left(\frac{4b}{\pi V}\right)s\right)} \cdot H_w(s)$$

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

Note The Von Kármán filter references refer to the same velocity transfer functions for both military specifications. The turbulence scale lengths changes between military references have not impacted the form of the turbulence velocity transfer functions.

Low-Altitude Model (Altitude < 1000 feet)

According to the military references, the turbulence scale lengths at low altitudes, where h is the altitude in feet, are represented in the following table:

MIL-F-8785C   **MIL-HDBK-1797**

$L_w = h$ $2L_w = h$

$L_u = L_v = \dfrac{h}{(0.177 + 0.000823h)^{1.2}}$ $L_u = 2L_v = \dfrac{h}{(0.177 + 0.000823h)^{1.2}}$

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

Von Karman Wind Turbulence Model (Continuous)

knots, for "moderate" turbulence the wind speed is 30 knots, and for "severe" turbulence the wind speed is 45 knots.

$$\sigma_w = 0.1 W_{20}$$

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

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**

$$L_u = L_v = L_w = 2500 \ ft \qquad\qquad L_u = 2L_v = 2L_w = 2500 \ ft$$

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:

Medium/High Altitude Turbulence Intensities (Probability of Exceedance)

Between Low and Medium/High Altitudes (1000 feet < Altitude < 2000 feet)

At altitudes between 1000 feet and 2000 feet, the turbulence velocities and turbulence angular rates are determined by linearly interpolating between the value from the low altitude model at 1000 feet transformed from mean horizontal wind coordinates to body coordinates and the value from the high altitude model at 2000 feet in body coordinates.

Von Karman Wind Turbulence Model (Continuous)

Dialog Box

Units

Define the units of wind speed due to the turbulence.

	Wind Velocity	Altitude	Air Speed
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:

Model	Description
Continuous Von Kármán (+q -r)	Use continuous representation of Von Kármán velocity spectra with positive vertical and negative lateral angular rates spectra.
Continuous Von Kármán (+q +r)	Use continuous representation of Von Kármán velocity spectra with positive vertical and lateral angular rates spectra.
Continuous Von Kármán (-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.

Von Karman Wind Turbulence Model (Continuous)

Model	Description
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

The first input is the altitude in units selected.

The second input is the aircraft speed in units selected.

The third input is a direction cosine matrix.

The first output is a three-element signal containing the turbulence velocities, in the selected units.

The second output is a three-element signal containing 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.

Von Karman Wind Turbulence Model (Continuous)

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

WGS84 Gravity Model

Purpose   Implement the 1984 World Geodetic System (WGS84) representation of Earth's gravity

Library   Environment/Gravity

Description

The WGS84 Gravity Model block implements the mathematical representation of the geocentric equipotential ellipsoid of the World Geodetic System (WGS84). The block output is the Earth's gravity at a specific location. Gravity precision is controlled via the **Type of gravity model** parameter.

The WGS84 Gravity Model block icon displays the input and output units selected from the **Units** list.

Dialog Box
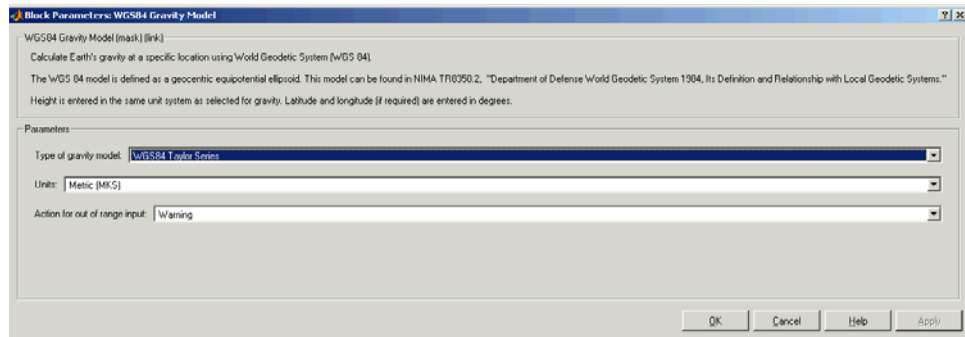
Type of gravity model

Specifies the method to calculate gravity:

- WGS84 Taylor Series
- WGS84 Close Approximation
- WGS84 Exact

Units

Specifies the input and output units:

	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 only available 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. In order 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 only available with **Type of gravity model WGS84 Close Approximation** or **WGS84 Exact**.

Month

Specifies the month used to calculate Julian Centuries from Epoch J2000.0.

This option is only available with **Type of gravity model WGS84 Close Approximation** or **WGS84 Exact** and only when Precessing reference frame is selected.

Day

Specifies the day used to calculate Julian Centuries from Epoch J2000.0.

This option is only available with **Type of gravity model WGS84 Close Approximation** or **WGS84 Exact** and only when Precessing reference frame is selected.

Year

Specifies the year used to calculate Julian Centuries from Epoch J2000.0. The year must be 2000 or greater.

This option is only available with **Type of gravity model WGS84 Close Approximation** or **WGS84 Exact** and only when Precessing reference frame is selected.

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 only available 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

The first input is a scalar containing the altitude in specified length units.

The second input is a scalar containing the latitude in degrees.

The third input is a scalar containing the longitude in degrees. This input is only available with **Type of Gravity Model WGS84 Close Approximation** or **WGS84 Exact**.

The output is a scalar value of gravity with the direction normal to the Earth's surface.

Assumptions and Limitations

The WGS84 gravity calculations are based on the assumption of a geocentric equipotential ellipsoid of revolution. Since the gravity potential is assumed to be the same everywhere on the ellipsoid, there must be a specific theoretical gravity potential that can be uniquely determined from the four independent constants defining the ellipsoid.

Use of the WGS84 Taylor Series model should be limited to low geodetic heights. It is sufficient near the surface when submicrogal precision is not necessary. At medium and high geodetic heights, it is less accurate.

Use of the WGS84 Close Approximation model should be limited to a geodetic height of 20000.0 m (approximately 65620.0 feet). Below this height, it gives results with submicrogal precision.

Examples

See the Airframe subsystem in the aeroblk_HL20 model for an example of this block.

References

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

Wind Shear Model

Purpose   Calculate wind shear conditions

Library   Environment/Wind

Description
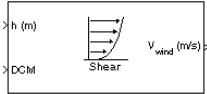
The Wind Shear Model block adds wind shear to the aerospace model. This implementation is based on the mathematical representation in the Military Specification MIL-F-8785C [1]. The magnitude of the wind shear is given by the following equation for the mean wind profile as a function of altitude and the measured wind speed at 20 feet (6 m) above the ground.

$$u_w = W_{20} \frac{\ln\left(\dfrac{h}{z_0}\right)}{\ln\left(\dfrac{20}{z_0}\right)}, \quad 3ft < h < 1000ft$$

where u_w is the mean wind speed, W_{20} is the measured wind speed at an altitude of 20 feet, h is the altitude, and z_0 is a constant equal to 0.15 feet for Category C flight phases and 2.0 feet for all other flight phases. Category C flight phases are defined in reference [1] to be terminal flight phases, which include takeoff, approach, and landing.

The resultant mean wind speed in the Earth-fixed axis frame is changed to body-fixed axis coordinates by multiplying by the direction cosine matrix (DCM) input to the block. The block output is the mean wind speed in the body-fixed axis.

Dialog Box
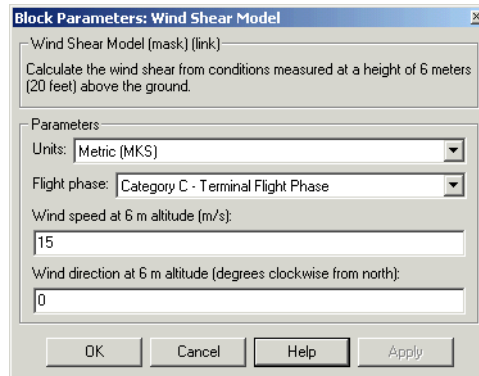
Units

Define the units of wind shear.

	Wind	**Altitude**
Metric (MKS)	Meters/second	Meters
English (Velocity in ft/s)	Feet/second	Feet
English (Velocity in kts)	Knots	Feet

Flight phase

Select flight phase:

- Category C — Terminal Flight Phases
- Other

Wind speed at 6 m (20 feet) altitude (m/s, f/s, or knots)

The measured wind speed at an altitude of 20 feet (6 m) above the ground.

Wind Shear Model

Wind direction at 6 m (20 feet) altitude (degrees clockwise from north)

The direction of the wind at an altitude of 20 feet (6 m), measured in degrees clockwise from the direction of the Earth x-axis (north). The wind direction is defined as the direction from which the wind is coming.

Inputs and Outputs

The first input is the altitude in units selected.

The second input is a 3-by-3 direction cosine matrix.

The output is a 3-by-1 vector of the mean wind speed in the body axes frame, in the selected units.

Examples

See the Airframe subsystem in the `aeroblk_HL20` model for an example of this block.

References

U.S. Military Specification MIL-F-8785C, 5 November 1980.

See Also

Discrete Wind Gust Model

Dryden Wind Turbulence Model (Continuous)

Purpose　　　Calculate the Earth's magnetic field at a specific location and time using the World Magnetic Model 2000 (WMM2000) block.

Library　　　Environment/Gravity
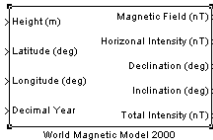
Description　The WMM2000 block implements the mathematical representation of the National Imagery and Mapping Agency (NIMA) World Magnetic Model 2000. The WMM2000 block calculates the Earth's magnetic field vector, horizontal intensity, declination, inclination, and total intensity at a specified location and time.

```
┌─────────────────────────────┐
│Height (m)     Magnetic Field (nT)│
│                                 │
│Latitude (deg) Horizonal Intensity (nT)│
│                 Declination (deg)│
│Longitude (deg)  Inclination (deg)│
│Decimal Year    Total Intensity (nT)│
└─────────────────────────────┘
   World Magnetic Model 2000
```

Dialog Box

World Magnetic Model 2000

Units

Specifies the input and output 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

The first input is the height, in selected units.

The second input is the latitude in degrees.

The third input is the longitude in degrees.

The fourth optional input is the decimal year.

The first output is the magnetic field vector in selected units.

The second optional output is the horizontal intensity in selected units.

The third optional output is the declination in degrees.

The fourth optional output is the inclination in degrees.

The fifth optional output is 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.

References   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/seg/WMM/DoDWMM.shtml

World Magnetic Model 2000

Aerospace Units

The main blocks of the Aerospace Blockset support standard measurement systems. The Unit Conversion blocks support all units listed in the following table.

Quantity	Metric (MKS)	English
Acceleration	meters/second^2 (m/s^2), kilometers/second^2 (km/s^2), kilometers/hour (km/h), kilometers/second (km/s)	inches/second^2 (in/s^2), feet/second^2 (ft/s^2), miles/hour (mph), miles/second (mps)
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)

Quantity	Metric (MKS)	English
Pressure	Pascal	pound/inch^2 (psi), pound/foot^2 (psf), atmosphere (atm)
Temperature	Kelvin, Celsius	Fahrenheit, Rankine
Velocity	meters/second (m/s), kilometers/second (km/s), kilometers/hour (km/h)	inches/second (in/sec), feet/second (ft/sec), miles/hour (mph), knots

Index