AeroSim

aeronautical simulation blockset

Version 1.2

User's Guide

Unmanned Dynamics

www.u-dynamics.com

AEROSIM BLOCKSET
Version 1.2
User's Guide

Unmanned Dynamics, LLC
No.8 Fourth St.
Hood River, OR 97031
(503) 329-3126
http://www.u-dynamics.com
info@u-dynamics.com

Contents

1 Introduction

The **AeroSim** aeronautical simulation blockset provides a complete set of tools for the rapid development of nonlinear 6-degree-of-freedom aircraft dynamic models. In addition to the basic aircraft dynamics blocks, the library also includes complete aircraft models which can be customized through parameter files.

Since the **AeroSim** blockset components are built using only basic **Simulink** blocks and portable C/C++ code, you can use **Real-Time Workshop** to automatically generate source code from your aircraft models.

Aircraft model examples include the Aerosonde UAV and the Navion general-aviation airplane.

The library allows importation of XML FlightGear aircraft configuration files (JSBSim format). Sample blocks for the Cessna 172, 182, and 310 based on JSBSim models can be found in the **FlightGear-Compatible** section of the **AeroSim** library.

1.1 Software Changes from Version 1.1 to 1.2

The following changes were done to existing library features:

- There are now multiple **FlightGear Interface** blocks, for each version of FlightGear supported by the Aerosim Blockset. We currently support FlightGear 0.9.2 and 0.9.4. For backward compatibility, the block name for FlightGear Interface for version 0.9.2 has been preserved.
- A bug regarding the calculation of the (2,2) element in the Direction Cosine Matrix from Euler Angles which existed in both the documentation and the Simulink implementation has been fixed, thus the DCM matrix is now computed correctly.
- The trim models for Aerosonde and Navion were generating errors due to the fact that the Wind Force and Wind Moment blocks have been removed in the previous release. The trim models have now been updated to the current implementation of wind effects, and they will run without errors.

The following new features were added to the **AeroSim** Blockset in this release:

- The **FlightGear Interface** and **Joystick Interface** S-functions are now fully reentrant - that is, multiple instances of each can be created and executed in the same Simulink model. The joystick interface function will take a new parameter - the Windows joystick ID, such that two instances of this function will sample separate joysticks.
- A new block has been added - the **R/C Transmitter Interface** for use with R/C radios (if interfacing hardware is present).

1.2 Software Changes from Version 1.01 to 1.1

The following changes were done to existing library features:

- The **FlightGear Interface** now uses the latest version of FlightGear which is 0.9.2. The interface S-function was updated to be able to connect to this version. The **C-172**, **C-182** and **C-310** aircraft models were updated to make use of the latest version of the JSBSim aircraft configuration files.

- The **Wind Force** and **Wind Moment** blocks were removed from the library; the wind effects are accounted for through the aerodynamic model. The only outputs from the wind model now are the wind linear and angular velocities.

- The **Total Acceleration** and **Total Moment** blocks do not take wind force and moment inputs anymore, since the wind effects have changed, as described in the item above.

- A bug in the **Inertia Coefficients** block resulted in the wrong value being computed for the inertia coefficient c_8. It is now corrected.

- The **Simple Aircraft** and **Glider** models were not running due to several variables being undefined. The models now run properly.

- A bug in the **ECEF Position** block caused incorrect transformation from geographic coordinates to Earth-centered Earth-fixed frame. The bug has been fixed.

1.3 Software Changes from Version 1.0 to 1.01

The following changes were done to existing library features:

- In the **Transformations** library folder, the block **Body-Inertial DCM** has been renamed to **Body-Inertial DCM From Quaternions**.

- In the **Body-Frame EOM** folder, the block **Kinematics** has been renamed to **Kinematics (Quaternions)**. Similarly, in the **Geodetic-Frame EOM**, the block **Attitude** has been renamed to **Attitude (Quaternions)**.

- In the **Complete Aircraft** folder, all of the aircraft models were internally updated to reflect the changes described above. If you have any Simulink models that contain any of these aircraft as library links, no modification are required on your Simulink diagrams. If you developed aircraft models independently using **AeroSim** components, you will need to update the links for the blocks described above.

The changes presented above were performed to allow development of equivalent attitude descriptions using Euler angles instead of quaternions.

1.4 New Features for Version 1.01

The following new features were added to the **AeroSim** Blockset in this release:

- Ability to import and use FlightGear aircraft dynamic models. FlightGear supports XML-based flight dynamic models in several formats. Currently, only the most popular format - JSBSim can be imported by the AeroSim Blockset. The FlightGear compatibility is provided through an xmlAircraft() parser script and through additional Simulink blocks provided in a new AeroSim library folder. The same folder contains 3 examples of aircraft models based on JSBSim configuration files - the Cessna 172, 182 and 310.

- Aircraft model trim and linearization script. The Matlab program trims an **AeroSim** aircraft model for a user-provided

flight condition and extracts the linear model at that flight condition.

- The addition of attitude representation by Euler angles. For this the following blocks were added to the library: in the **Transformations** section - **Body-Inertial DCM From Euler Angles**, **Quaternions From Euler Angles**, in the **Body-Frame EOM** section - **Kinematics (Euler Angles)**, and in the **Geodetic-Frame EOM - Attitude (Euler Angles)**.

1.5 Keeping up-to-date with the AeroSim Blockset: The AeroSim Mailing List

The **AeroSim** Blockset is evolving steadily. An AeroSim mailing list was created at Yahoo Groups for new release announcements as well as for technical discussions regarding the AeroSim Blockset. The mailing list can be found at:
http://groups.yahoo.com/group/aerosim/

We encourage our users to subscribe to this list, by sending a blank email to:
aerosim-subscribe@yahoogroups.com.

1.6 System Requirements

The minimum software requirements for the **AeroSim** blockset are **Matlab** version **6** and **Simulink** version **4**. The blockset will not operate properly with earlier versions of **Matlab** (such as **5.x**) since **AeroSim** makes use of matrix signals and matrix operations which are not available in the earlier versions of **Simulink**.

The Flight Simulator and FlightGear interface blocks which can be used for visual output require **Microsoft Flight Simulator 2000** or later, respectively **FlightGear** version **0.8**.

1.7 Contents of the installation CD

The installation CD includes the following:

1. The AeroSim installer, in the directory **aerosim-setup**.

2. The FlightGear Flight Simulator - an open-source flight simulator application, in the directory **flightgear**. The latest version of the software can also be downloaded from:
http://www.flightgear.org.

3. Utilities for Microsoft Flight Simulator, in the directory **fsutil**. These are the following:

   - Flight Simulator Update 2b, for Flight Simulator 2000 only. This is required for using the other utility programs with Microsoft Flight Simulator 2000.
   - FSUIPC, an utility which allows inter-process communication with Microsoft Flight Simulator 2000 and 2002 (by Peter L. Dowson). The latest version of this application can be downloaded from
http://www.flightsim.com.
   - WIDEFS, an utility which extends FSUIPC functionality by allowing inter-process communication with Flight Simulator over the local network (by Peter L. Dowson). Also, the latest version of the program can be

downloaded at
http://www.flightsim.com.

- Flight Simulator aircraft. Visual models for the Aerosonde UAV and for the Navion, the two aircraft examples featured in the **AeroSim** blockset demos can be found on the installation disk, in the **/fsutil/aircraft/** directory. There are three versions of each aircraft model in standard FS5 format, in Flight Simulator 2000 format, and in Flight Simulator 2002 format. The standard FS5 models can be used with any Flight Simulator version from 5.0 up to 98, by making use of the **Aircraft Converter** included with your Flight Simulator software. The FS2000 and FS2002 aircraft can be installed by directly copying the airplane directories as provided on the **AeroSim** disk to your **/$FLIGHTSIM/aircraft/** directory.

The software available on the installation CD can also be downloaded directly from the **AeroSim** product page at:
http://www.u-dynamics.com/aerosim/.

1.8 Install the AeroSim Blockset

To install the **AeroSim** blockset, run the **setup.exe** executable which is located in the **aerosim-setup** directory of the installation CD. By default, the installer will place the AeroSim files in the **C:/Program Files/AeroSim** directory. A dialog box will give the user the option to choose a different directory for installation. The setup program will also create a new Start Menu entry, in which it will place a link to a PDF copy of this User Guide.

After the installer finishes copying all the files it will update the **Matlab** path file by adding entries for the **AeroSim blockset**. The path file that will be modified is **$MATLABROOT/toolbox/local/pathdef.m**. Before making any changes, a back-up copy of the original file will be saved as **pathdef.old**. The entries that will be added to the path listing are **$AEROSIMROOT**, **$AEROSIMROOT/samples**, **$AEROSIMROOT/trim**, **$AEROSIMROOT/util**, and **$AEROSIMROOT/fgutil**. If for some reason the path file cannot be edited by the installer (current user does not have write permission, or file does not exist) the installation program will display a warning message.

After completing the **AeroSim** blockset installation, start **Matlab** and open the **Simulink** browser window. The **AeroSim** blockset will be available for use, and the library tree will look similar to Fig. 1.

1.9 Uninstall the AeroSim Blockset

To remove the **AeroSim Blockset** from your machine, choose **UnInstall AeroSim** from the Windows Start Menu, or uninstall the program from the Windows Control Panel. The uninstaller will remove all files in the AeroSim directory, and the Start Menu entries. However, the uninstaller is unable to undo the changes made to the Matlab path file. You must manually replace your current **pathdef.m** file with the original file which was saved as **pathdef.old**, otherwise Matlab will display a warning during startup.

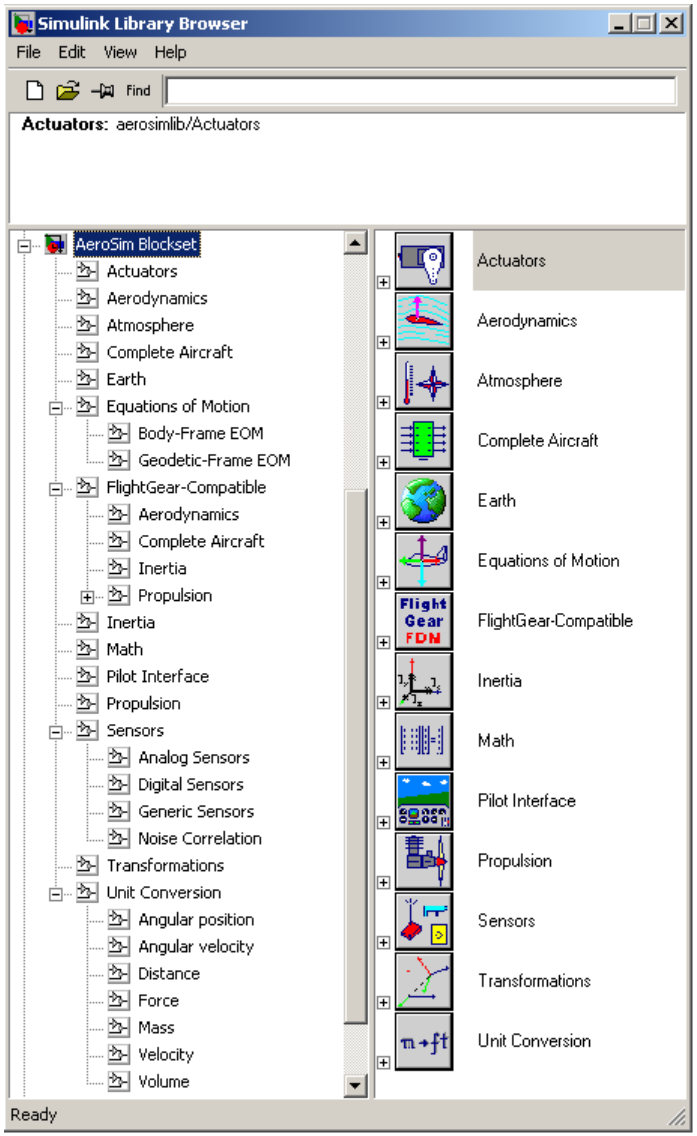


Figure 1: Simulink Library Browser

1.10 Library Description

The **AeroSim** library folders, presented in Fig. 2, provide more than one-hundred blocks commonly used in the development of aircraft dynamic models. These include nonlinear equations of motion, linear aerodynamics, piston-engine propulsion, aircraft inertia parameters, atmosphere models, Earth models, sensors and actuators, frame transformations, and pilot interfaces such as joystick input and 3-D visual output.

The library also provides complete aircraft models built using **AeroSim** blocks, and which can be customized to particular aircraft by editing an aircraft parameter file.

In addition to the block library, a set of Simulink demos can be found in the **samples** directory of the **AeroSim** blockset. These provide dynamic models of actual aircraft such as the **Aerosonde** Unmanned Air Vehicle and the **Navion** general-aviation airplane. The demo models are presented in detail in the next section.

Aircraft model trim and linearization functions as well as trim examples for **Aerosonde** and **Navion** can be found in the **trim** directory.

The **fgutil** directory contains XML parser scripts that can load JSBSim aircraft configuration files in Matlab aircraft structures.

The **util** directory contains several useful engineering math functions including Euler and Quaternion attitude representations, and eigenvalue analysis.

The **src** directory provides the C/C++ source code for the S-functions that are implemented as C-MEX files. This allows you to use Real-Time Workshop with any aircraft models developed with AeroSim blocks and be able to build the resulting source code for other hardware platforms.

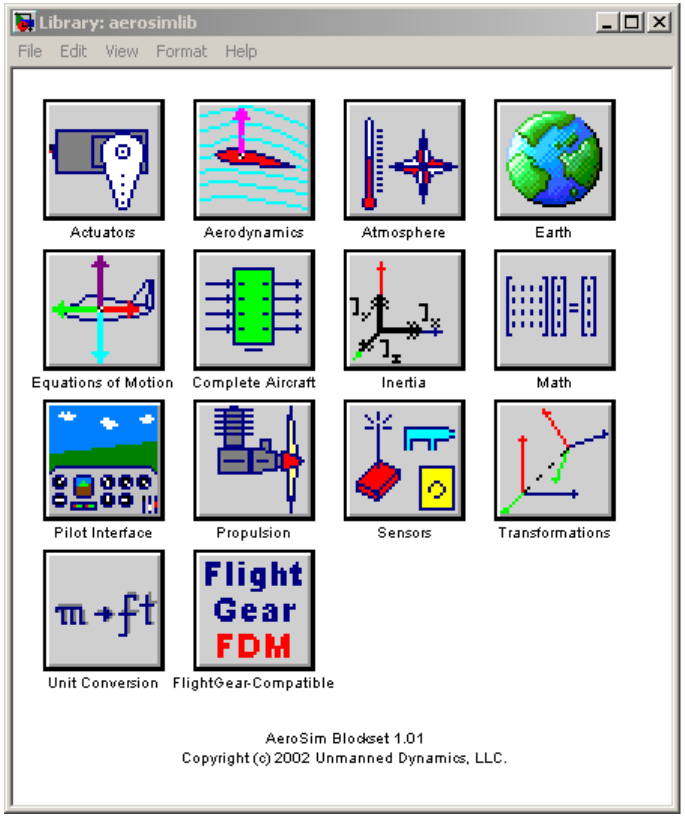


Figure 2: AeroSim Library

2 Aircraft Model Demos

The following Simulink models can be found in the **samples** directory of the AeroSim library. They illustrate the functionality of the AeroSim blockset. The models presented next are using the Aerosonde for aircraft dynamics, but similar models exist for the Navion airplane as well.

2.1 Open-loop Flight

The first demo, with the Simulink model shown in Fig. 3, represents a simulation of the aircraft in open-loop flight. That is, all aircraft control inputs are set to fixed values, independent of the aircraft states. The Simulink model can be opened by typing its name

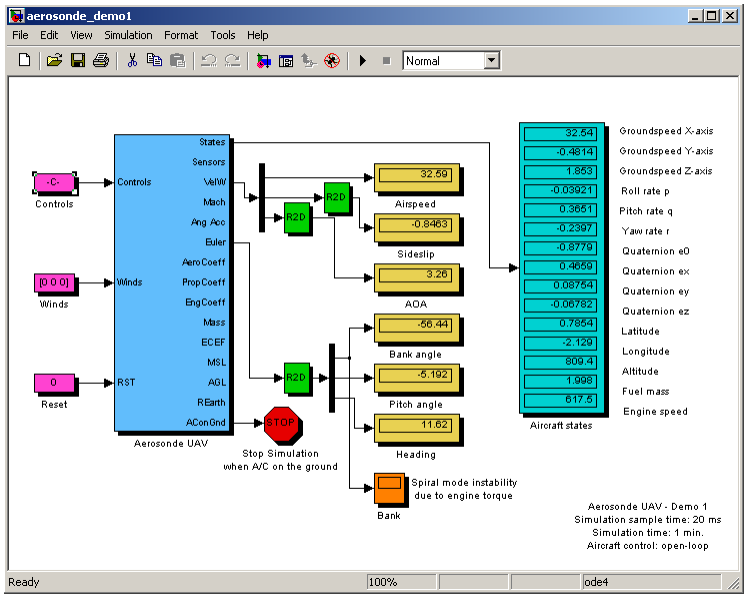


Figure 3: Aerosonde open-loop flight demo

at the Matlab commmand prompt: **aerosonde_demo1**. The aircaft model used for Aerosonde is the **6-DOF Aircraft Model (with Body-frame EOM)** from the **AeroSim** blockset. The aircraft parameter file used by the Aerosonde model is **aerosondecfg.mat** and it was generated using the Matlab script **aerosondeconfig.m**. Both files are located in the sub-directory **samples** of the main AeroSim directory.

The unbalanced roll moment caused by the propulsion system excites the spiral mode and the aircraft settles in a constant bank angle turn, as shown in Fig. 4.

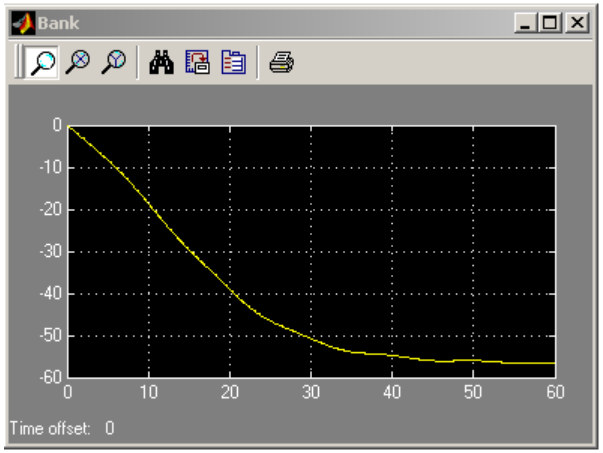


Figure 4: Bank angle output

2.2 Lateral Control

To open the second demo, type **aerosonde_demo2** at the Matlab command prompt. The Simulink diagram is shown in Fig. 5. In

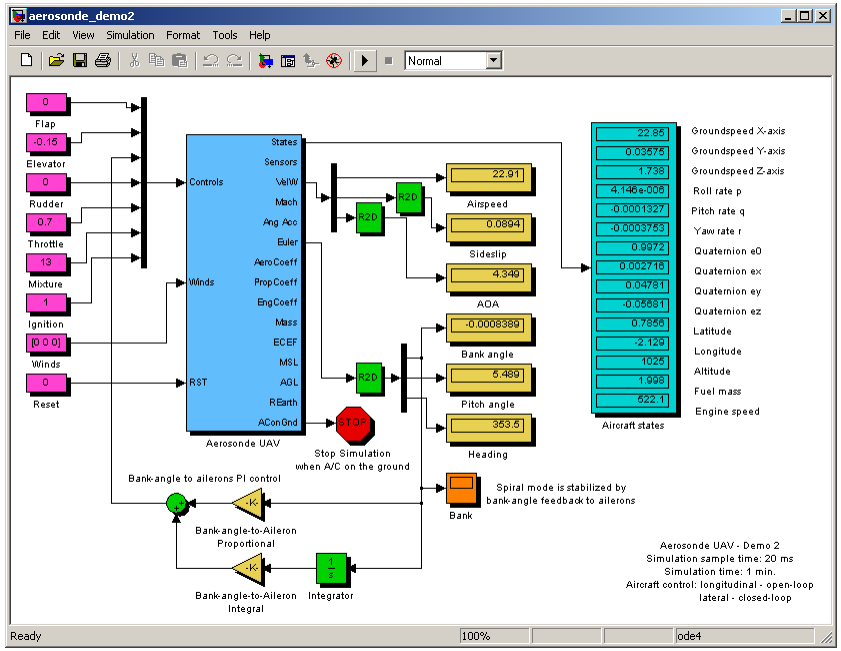


Figure 5: Aerosonde lateral control demo

this example, we stabilize the lateral dynamics of the aircraft by adding a wing leveler. This is implemented using proportional and integral feedback from bank angle to ailerons. We can see in Fig. 6 that the bank angle now settles to zero (wings level attitude).

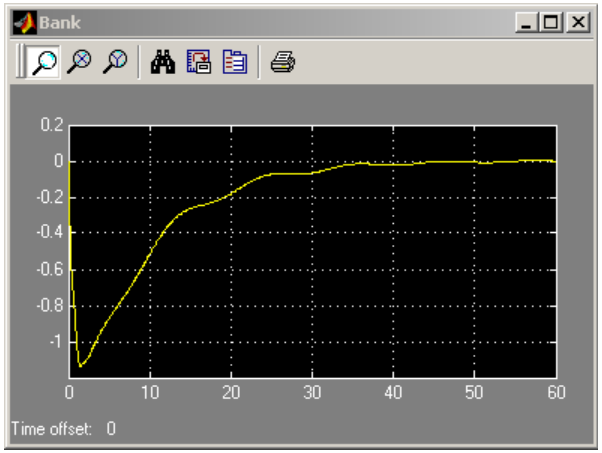


Figure 6: Bank angle is now stabilized

2.3 Airspeed Control

To open the third demo, type **aerosonde_demo3** at the Matlab command prompt. The Simulink diagram is shown in Fig. 7. With

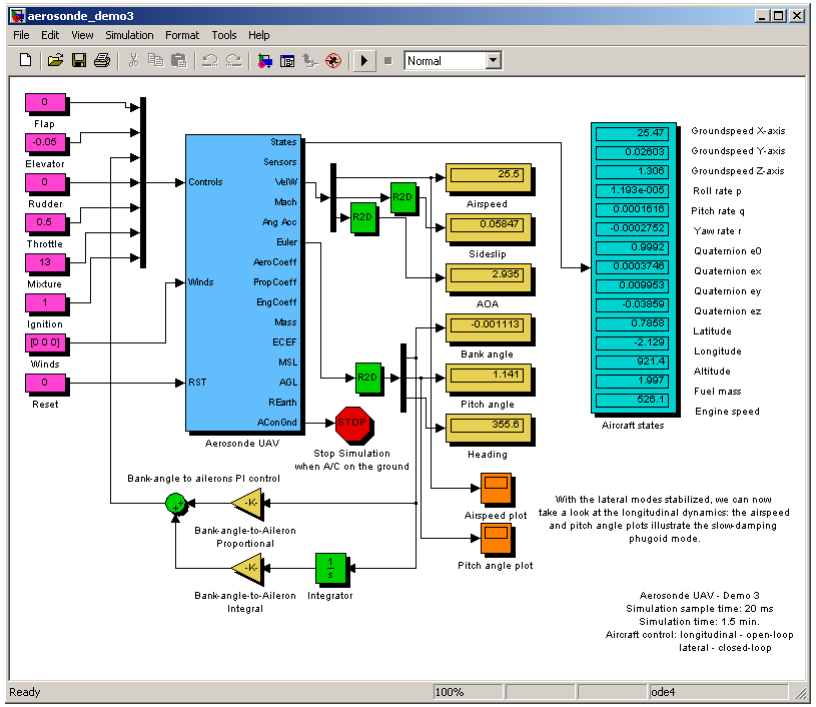


Figure 7: Aerosonde open-loop flight demo

the lateral dynamics of the aircraft now under control, we take a quick look at the longitudinal dynamics. In Fig. 8 we can see the time history of airspeed and pitch angle. The initial phugoid oscillations take approximately 60 seconds to damp out. At the same time, the airspeed settles to a value which depends on the elevator setting. The model **aerosonde_demo4** adds an airspeed control loop using PID (proportional, integral, derivative) control laws - see Fig. 9. The airspeed command is set to a constant value of 25 m/s. By running the simulation, you should see a plot of the airspeed and pitch angle similar to Fig. 10. The airspeed does indeed settle to the commanded value, while the phugoid oscillations are virtually eliminated and the pitch angle settles to a value of approximately 1.7 degrees.

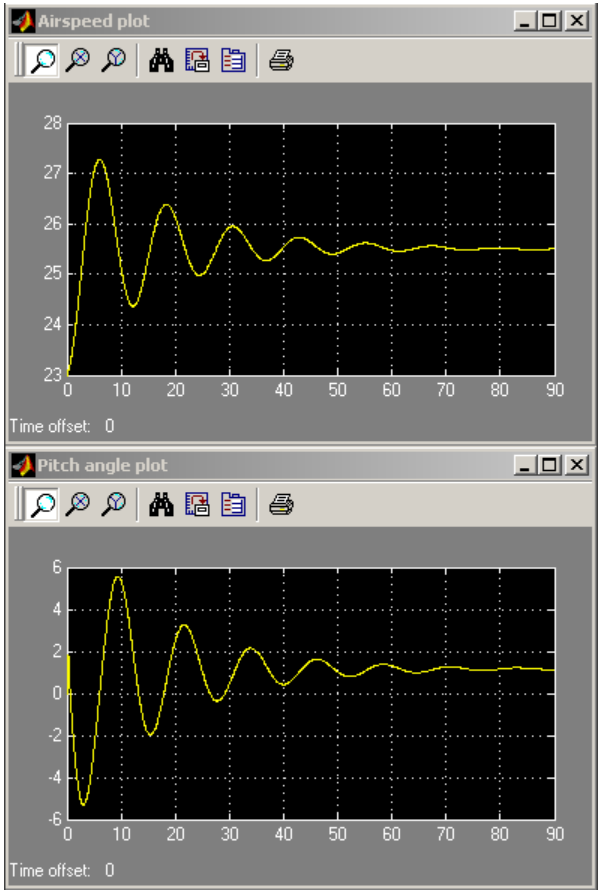


Figure 8: Airspeed and pitch angle outputs

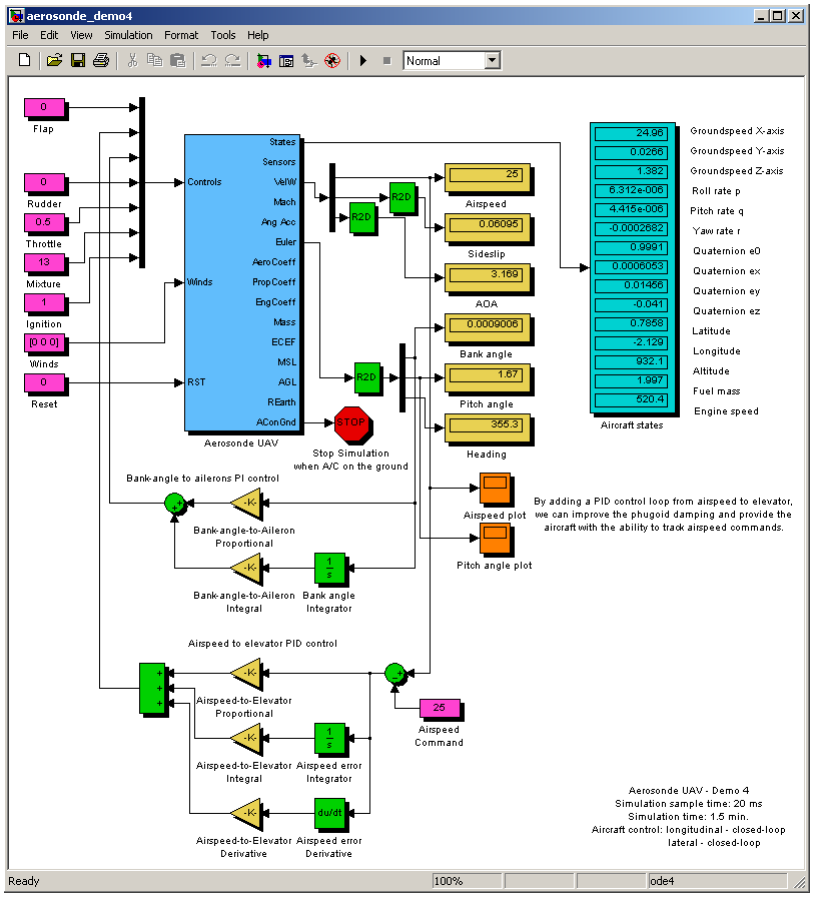


Figure 9: Aerosonde longitudinal control demo

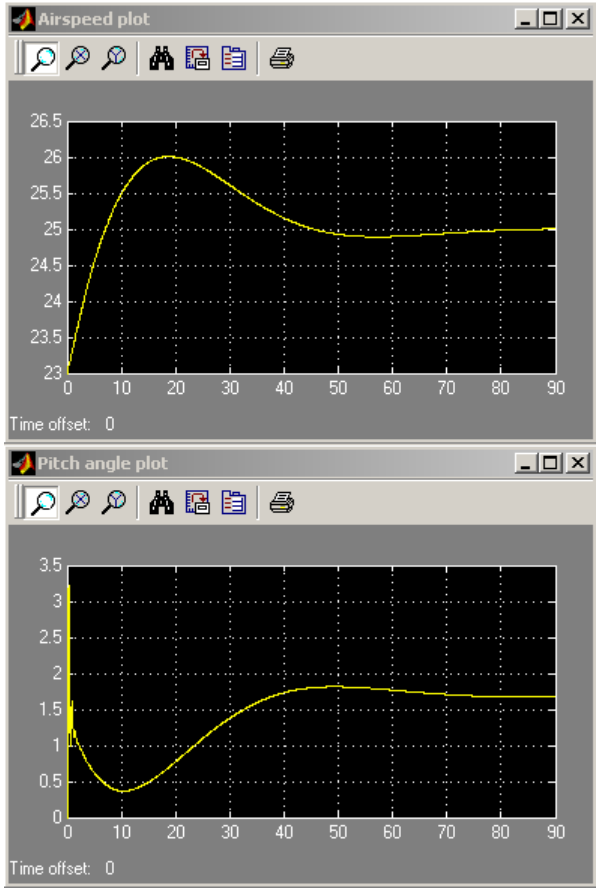


Figure 10: Airspeed and pitch angle are now stabilized

2.4 Wind Effects

To open the fifth demo, type **aerosonde_demo5** at the Matlab command prompt. The Simulink diagram is shown in Fig. 11. This example illustrates the effects of wind. The background wind velocity components chosen for this simulation are 25 m/s from South and 10 m/s from East - resulting in a wind speed greater than the aircraft speed (regulated at 25 m/s by the airspeed control loop).

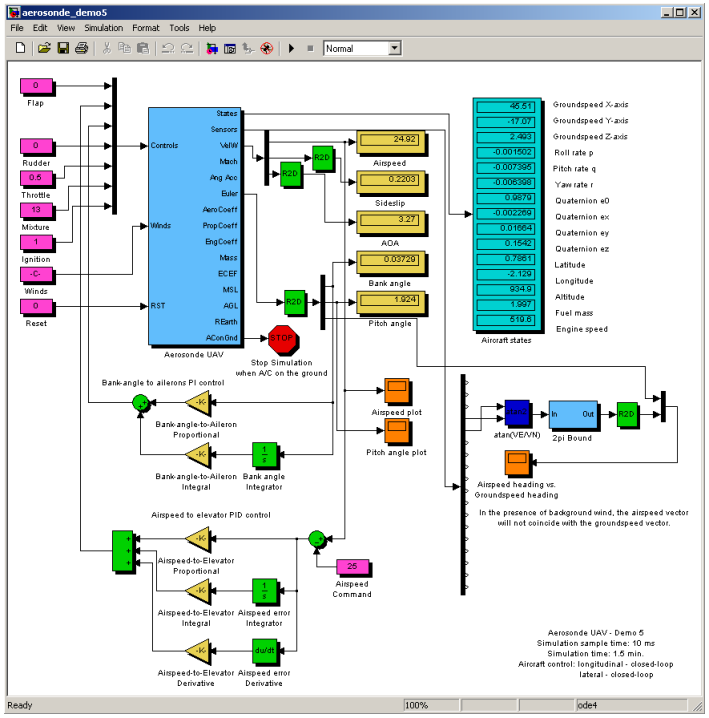


Figure 11: Aerosonde wind demo

First, we notice that compared to the results of previous simulations, the airspeed and pitch plots are noisy - the effect of the von Karman turbulence models - see Fig. 12.

We also plot the orientation of the aircraft airspeed vector, versus the orientation of the aircraft groundspeed vector. When there is no wind the two vectors are overlapped, and their magnitudes are identical. In the presence of background wind, this will cause a difference in the orientation and the magnitude of the airspeed and groundspeed vectors. We can see that in our case, there is a difference of approximately 30 degrees (airspeed heading is 25, while the groundspeed heading is 355).

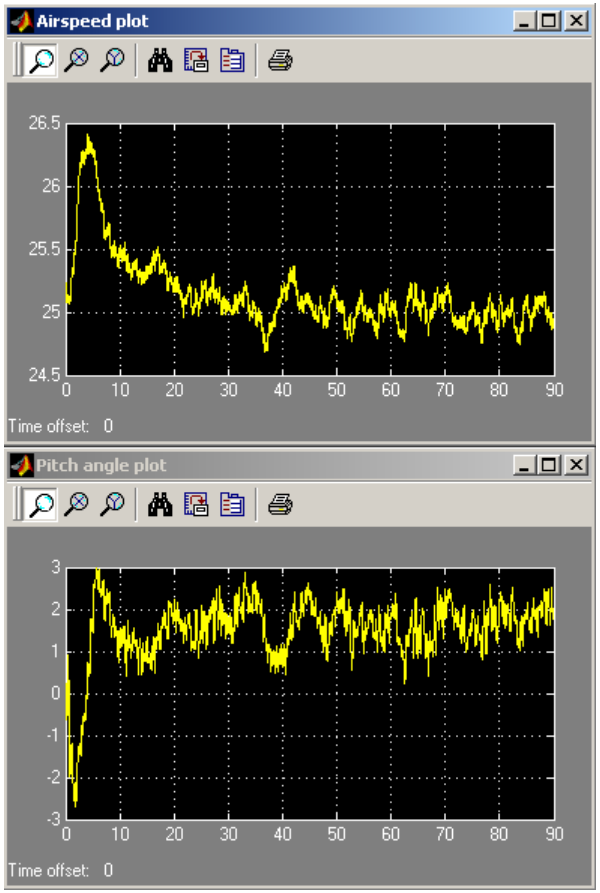


Figure 12: Airspeed and pitch angle in the presence of atmospheric turbulence

2.5 Inertial Navigation

To open the sixth demo, type **aerosonde_demo6** at the Matlab command prompt. The Simulink diagram is shown in Fig. 13. This example is using the **Inertial Navigation System** block from

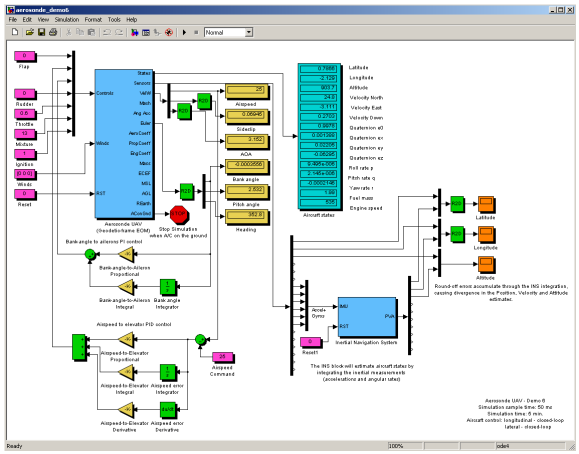


Figure 13: INS demo

the **AeroSim** library to integrate the inertial sensor measurements provided by the aircraft model. The inertial measurements are integrated to obtain estimates of the aircraft states (position, velocity, and attitude). These are plotted against the aircraft states provided by the aircraft model - Fig. 14. We can see that the INS estimates diverge from the real states slowly in time. Although the model assumes perfect sensors, small errors such as those induced by the fixed-step integration scheme accumulate during the INS integration process causing this divergent behavior. PVA estimates can be improved by adding a navigation Kalman filter.

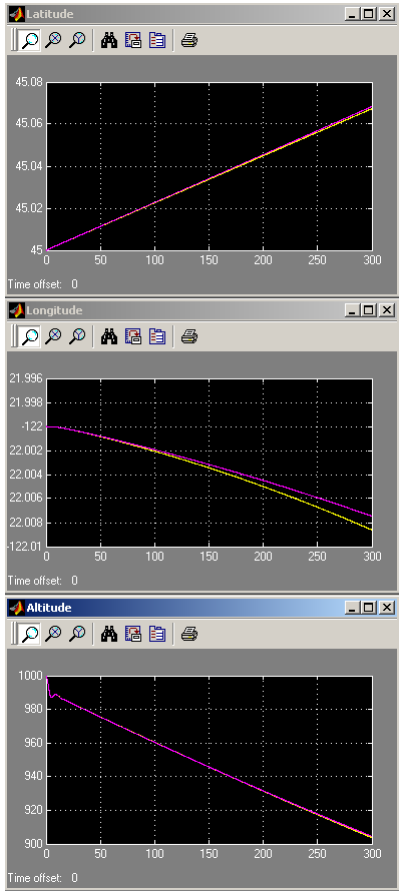


Figure 14: Aircraft position vs. INS position estimate

2.6 Joystick Control

The joystick demo, with the Simulink model **joystick_demo1** requires the presence of a properly calibrated joystick on the system. The model is shown in Fig. 15. While the simulation is running,

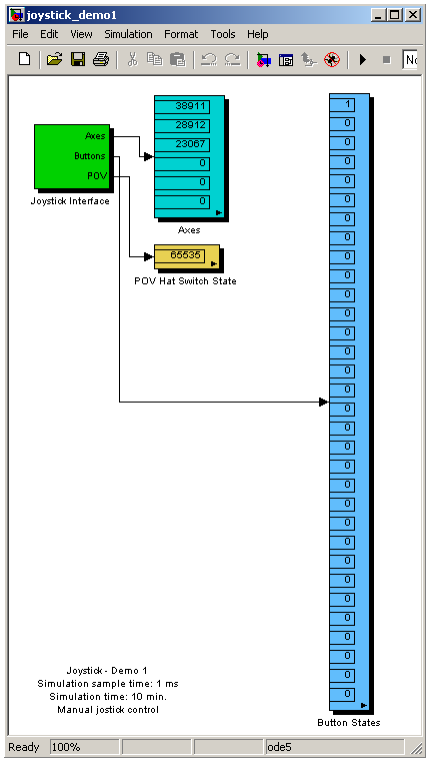


Figure 15: Joystick demo

the display blocks will show the current values for the joystick position on the 6 axes, as well as the buttons and POV hat switch states. Any joystick inputs given by the user will be displayed in real-time.

2.7 Interfacing to FlightGear Flight Simulator

This demo illustrates the FlightGear interface functionality. First, open the Simulink model **aerosonde_demo_fgfs**. The Simulink model, shown in Fig. 16, can be executed, regardless of whether FlightGear is installed or not on the computer. The demo model will dump aircraft state data over the local network connection using the UDP protocol. Double-click on the **FlightGear Interface** block. This will open the parameter dialog, as shown in Fig. 17. In the upper edit box, type the name of the computer on which FlightGear is installed. The parameter should be a string (single quotation marks). If it is the same computer as the one running the Matlab/Simulink environment, then just leave the name as **'localhost'**. However, for good performance we recommend keeping the Matlab and the FlightGear applications on separate machines, since both applications put a significant load on the CPU. The lower edit box allows the user to specify the sample time at which aircraft state data will be sent to FlightGear. Due to Simulink constraints, this should be a multiple of the base model sample time. For this demo it is probably best to leave this setting unchanged.

First start FlightGear Flight Simulator, by using the external flight model option. This can be done with a batch file, or directly from the command prompt by typing the following command from the main FlightGear directory:

```
bin\fgfs --native-fdm=socket,in,30,,5500,udp --fdm=external
```

With these parameters, FlightGear will disable its internal flight dynamics, and it will accept aircraft states from the local network, as a UDP stream, on port 5500.

Now run the Simulink model **aerosonde_demo_fgfs**. The FlightGear screen should display a view from the aircraft in straight and

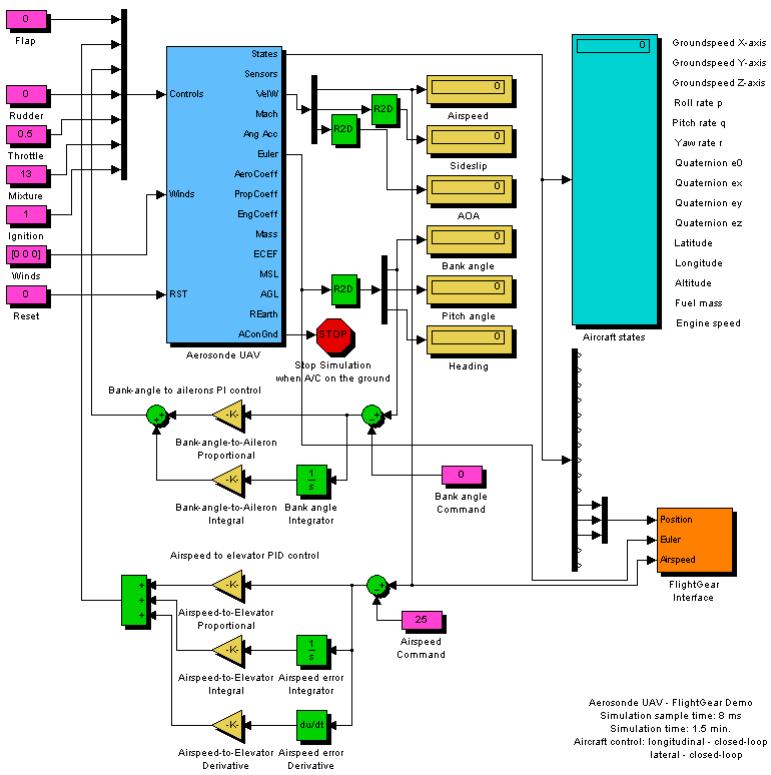


Figure 16: FGFS interface demo

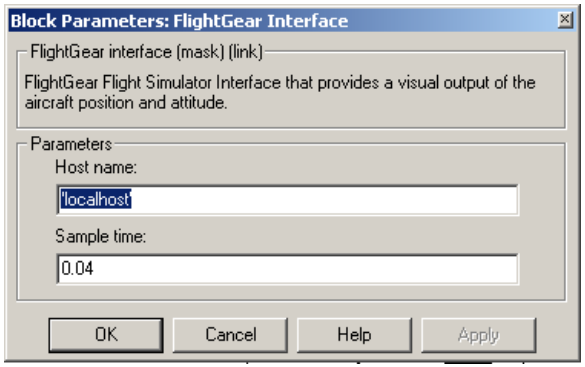


Figure 17: FGFS interface parameters

Figure 18: FGFS shows aircraft in straight and level flight

level flight, see Fig. 18. While the simulation is running, change the value in the **Bank angle Command** block from 0 to 30 degrees. The FlightGear screen will show the aircraft banking right, as it settles at the new bank angle command - see Fig. 19.

Figure 19: FGFS shows aircraft in 30-deg. bank angle turn

2.8 Interfacing to Microsoft Flight Simulator

This demo will present the methodology for interfacing an aircraft dynamic model running in Simulink to a computer running **Microsoft Flight Simulator**.

1. If using **Flight Simulator 2000**, install the latest patch (update 2b).

2. Unzip **fsuipc.zip** in a temporary directory and follow the installation instructions.

3. Unzip **widefs.zip** in a temporary directory and follow the installation instructions.

4. Check to make sure FSUIPC is installed by starting Flight Simulator and looking for FSUIPC in the Modules pull-down menu, as in Fig. 20.

5. Check if the WideFS server is running by looking at the Flight Simulator window title-bar - see Fig. 20.

6. On the Matlab machine, copy the WideFS client files (WideClient.exe and WideClient.ini). Edit the client initialization file to specify the address of the computer running the Flight Simulator with WideServer.

7. Launch the WideFS client program. If properly configured, the client should connect in a couple of seconds to the server.

8. On the Matlab machine, open the Simulink demo **aerosonde_demo_msfs.mdl**. The demo model is shown in Fig. 21

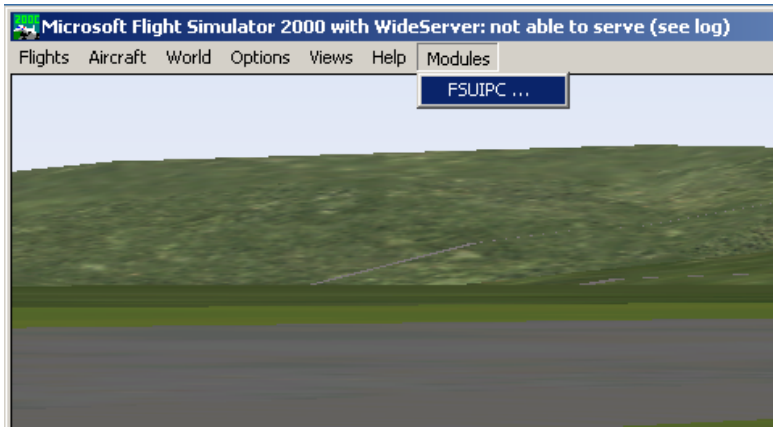


Figure 20: FSUIPC in the Modules pull-down menu

9. Start the Simulink model. The Flight Simulator screen will display the current position and attitude of the aircraft. Give the autopilot a new bank angle command, by updating the **Bank angle Command** block. You should be able to see the change in aircraft attitude in Flight Simulator immediately.

   Particular aircraft visual models can be loaded in **Microsoft Flight Simulator** before starting the simulation. The FS models provided on the **AeroSim** installation CD include the Aerosonde - Fig. 22 and the Navion - Fig. 23.

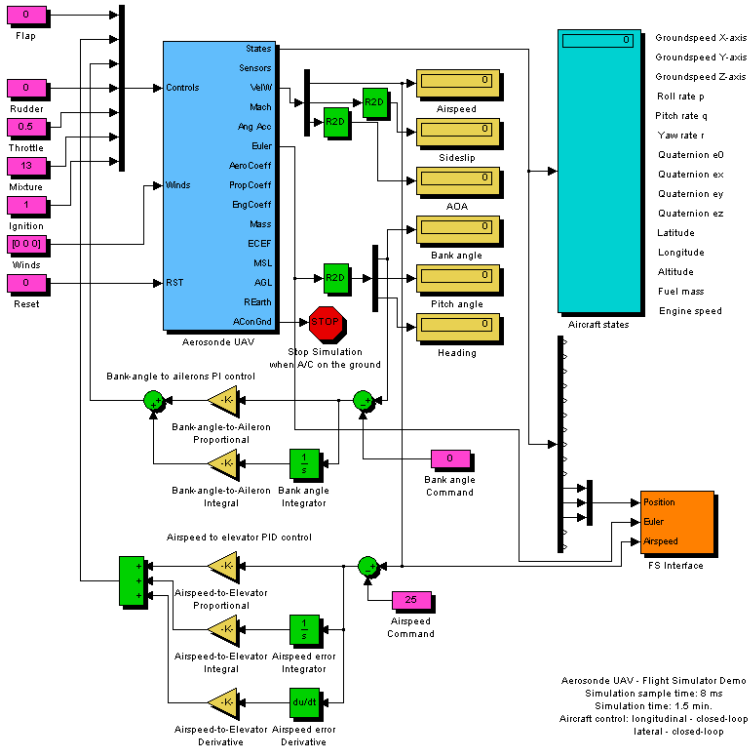


Figure 21: Flight Simulator Demo Model

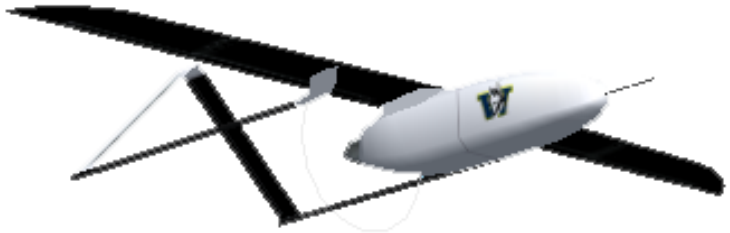


Figure 22: MSFS visual model of Aerosonde

Figure 23: MSFS visual model of Navion

2.9 FlightGear Aircraft Demos

The complete models for several aircraft that use **JSBSim** configuration files can be found in the AeroSim library. Several ready-to-run demos are also provided in the **samples** directory. These are: **c172fg_demo.mdl**, **c182fg_demo.mdl**, and **c310fg_demo.mdl** for the Cessna-172, 182, and 310 respectively.

Before running any of these Simulink models, make sure to set the correct FlightGear path string in the aircraft model block parameters, by double-clicking on the aircraft block and editing the parameter *FlightGear path*.

2.10 Aircraft Model Trim and Linearization

Two aircraft trim and linearization examples can be found in the **trim** directory - for the Aerosonde UAV and for the Navion airplane. Each of the examples includes a Simulink model of the aircraft and a Matlab script which performs the trim and the linearization of the aircraft dynamics.

For the Aerosonde UAV, the Simulink model is **aerosonde_trim.m** and the trim program is **trim_aerosonde.m**. The Simulink model is shown in Fig. 24.

The attitude equations in the aircraft model are implemented using the Euler angle representation which provides more intuitive trim results. Several aircraft control inputs - the flap, the mixture and the ignition are assumed constant through-out the trim procedure. The values for these constants, as well as the aircraft model block parameters are read from a Matlab structure created by the trim program (TrimParam).

The aircraft trim parameter structure has the following elements:

- SampleTime = the sample time at which the Simulink model will run during the trim process.
- FinalTime = the final time for the Simulink mode during the trim process.
- SimModel = a string with the name of the Simulink model to be trimmed (in our case 'aerosonde_trim').
- VelocitiesIni = the initial value for the aircraft velocity integrators (Force Equations).

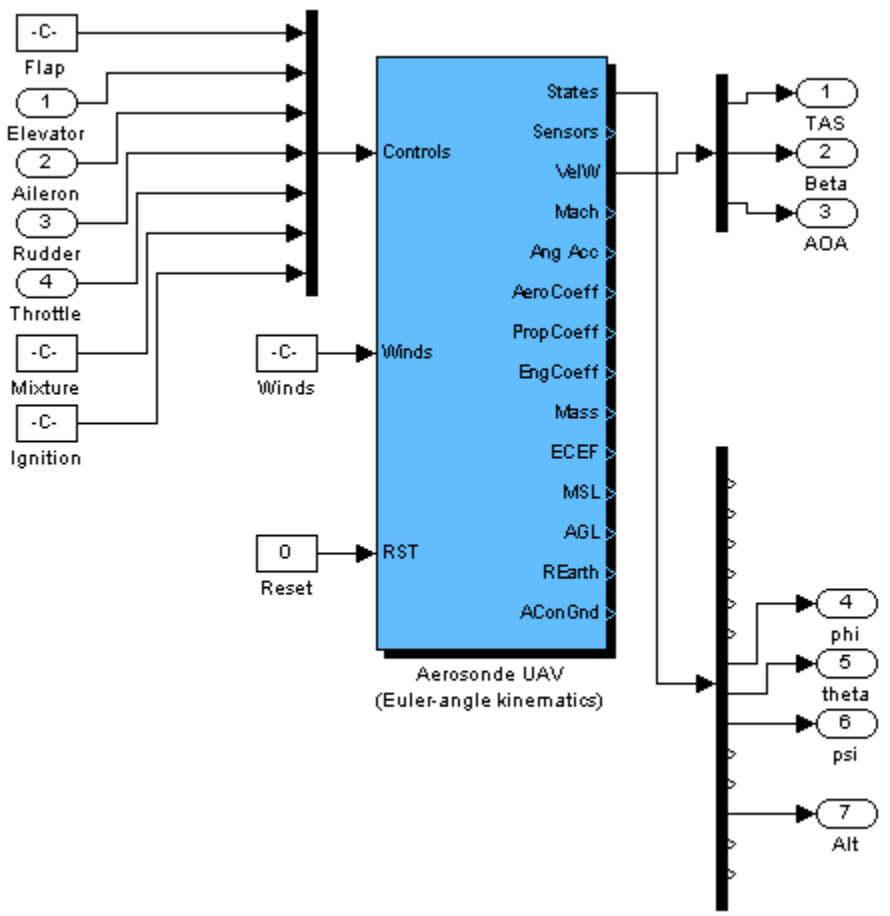


Figure 24: Simulink Model for Trim and Linearization

- RatesIni = the initial value for the aircraft angular rate integrators (Moment Equations).
- AttitudeIni = the initial value for the Euler angle integrators (Kinematic Equations).
- PositionIni = the initial value for the geographic position integrators (Navigation Equations).
- FuelIni = the initial value for the fuel mass integrator.
- EngineSpeedIni = the initial value for the engine rotation speed.
- Airspeed = the airspeed at which the aircraft will be trimmed.
- Altitude = the altitude at which the aircraft will be trimmed.
- BankAngle = the bank (roll) angle at which the aircraft will be trimmed.
- Elevator = the initial guess for the elevator position at trim condition.
- Aileron = the initial guess for the aileron position at trim condition.
- Rudder = the initial guess for the rudder position at trim condition.
- Throttle = the initial guess for the throttle position at trim condition.
- Flap = the flap setting at which the aircraft will be trimmed.
- Mixture = the air/fuel mixture at which the aircraft will be trimmed.
- Ignition = choose whether this is a power-on or power-off trim condition.
- Winds = the background wind vector for which the aircraft will be trimmed.
- StateIdx = the state order index maps the order of states in the Simulink diagram to a desired order of states. Often the order in which Matlab considers the states in a Simulink block diagram is different that the order we intend to use. In this case, the desired order is [Velocities Rates Attitude Position Fuel EngineSpeed].
- Options = the Matlab structure which defines the optimization function options.
- SimOptions = the Matlab structure which defines the simulation function options.
- NAircraftStates = the size of the desired aircraft state vector.
- NSimulinkStates = the size of the Simulink model state vector (could be larger since it could include states for other dynamics which are of no interest for trim (such as the turbulence filter states)

The trim and linearization script will create the trim parameters structure with values provided by the user and with default values for several parameters.

The script performs several steps, as presented below:

- Set the initial parameters. These include the Simulink model name and sample time settings as well as default values for aircraft control inputs.

- Identify the order of the aircraft states. The identification is done by running the Simulink model for a single sample time with some default initial guesses and reading back the Simulink model state vector to identify the indices for each of the states of interest.

- Initial guess for the aircraft controls. The procedure first requires user input regarding the flight condition at which the aircraft is to be trimmed. Once the flight condition is completely defined, the Simulink model is run for a limited amount of time, in an iterative process, and the aircraft control inputs are adjusted each time by proportional feedback from selected model outputs (for example, elevator feedback is provided by airspeed error, throttle feedback - by altitude error, and aileron feedback - by bank angle error). The method provides a better initial guess for the next step - the optimization, but the feedback gains are aircraft-specific, therefore the trim script requires some adjustments for each new aircraft.

- Perform the aircraft trim. Once a reasonable initial guess for the aircraft control inputs is obtained, the program will run the optimization which will accurately trim the aircraft for the selected flight-condition. The Matlab function used by this procedure is *trim*.

- Extract the linear model. Knowing the trim inputs and states, the nonlinear aircraft model is linearized about the trim condition, using the Matlab function *linmod*. The resulting linear model is then decoupled into longitudinal and lateral-directional plants; this procedure is valid only if the aircraft is trimmed for straight and level flight. At the end of the linearization procedure, a simple eigenvalue analysis of the longitudinal and lateral-directional dynamics is performed.

To illustrate the process, we will trim the Aerosonde model for a typical flight condition ($V_a = 23m/s$, $h = 200m$, $\phi = 0$). By running the trim program **trim_aerosonde.m** we obtain the following output:

```
Setting initial trim parameters...

The Simulink modelaerosonde_trim.mdl will be trimmed.

Identifying the order of the Simulink model states...done.

 Choose flight condition:
--------------------------
Trim airspeed [m/s]: 23

Trim altitude [m]: 200

Trim bank angle[rad]: 0

Fuel mass [kg]: 2

Flap setting [frac]: 0

Computing the initial estimates for the trim inputs...

Iteration # 1, Airsp err =   3.86 m/s, Alt err =   -23.96 m, phi
err = -15.80 deg.

Iteration # 2, Airsp err =   0.88 m/s, Alt err = 12.95 m, phi err
= -24.39 deg.
```

```
Iteration # 3, Airsp err =   0.27 m/s, Alt err = 2.89 m, phi err =
10.67 deg.

Iteration # 4, Airsp err = 0.04 m/s, Alt err =     2.14 m, phi err
=   1.04 deg.

Iteration # 5, Airsp err =   0.05 m/s, Alt err = 0.85 m, phi err =
1.07 deg.

Iteration # 6, Airsp err =   0.04 m/s, Alt err = 0.35 m, phi err =
0.46 deg.

Iteration # 7, Airsp err =   0.03 m/s, Alt err = 0.16 m, phi err =
0.19 deg.

Iteration # 8, Airsp err = 0.02 m/s, Alt err =     0.10 m, phi err
=   0.07 deg.

Done.

Initial guesses for trim inputs are:
   Elevator = -0.1034
   Aileron = -0.0093
   Rudder = 0.0000
   Throttle = 0.4925

Performing the aircraft trim...

f-COUNT     MAX{g}         STEP Procedures
   20     0.567619            1
   40    0.0781559            1
   60    0.0581701            1   Hessian modified twice
   80     0.521641            1   Hessian modified
  100    0.0379462            1   Hessian modified
  120     0.130393            1   Hessian modified twice
  140  0.000300865            1   Hessian modified
  160  0.000299109            1   Hessian modified twice
  180  0.000299108            1   Hessian modified twice
  200  0.000299153            1   Hessian modified
  220  0.000299106            1   Hessian modified twice
  240  0.000299116            1   Hessian modified twice
  260  0.000299417            1   Hessian modified twice
  280  0.000299103            1   Hessian modified twice
  300  0.000299113            1   Hessian modified twice
  320   0.00203574            1   Hessian modified twice
  343   0.00221129        0.125   Hessian modified twice
  363  0.000299534            1   Hessian modified twice
  364   0.00029908            1   Hessian modified twice
Optimization Converged Successfully Active Constraints:
     1
     2
     3
     4
     5
     6
     7
     8
     9
    10
    11
    19
    28
    30

Finished. The trim results are:
INPUTS:
   Elevator = -0.1088
   Aileron = -0.0084
   Rudder = -0.0010
   Throttle = 0.4846
STATES:
   u =  22.95 m/s
   v =   0.01 m/s
   w =   1.51 m/s
   p =  -0.00 deg/s
   q =  -0.00 deg/s
   r =  -0.00 deg/s
   phi =  -0.02 deg
   theta =   3.77 deg
   psi =   0.02 deg
   Alt = 200.00 m
   Fuel =   2.00 kg
   Engine =   4812 rot/min
OUTPUTS:
   Airspeed =  23.00 m/s
   Sideslip =   0.02 deg
   AOA =   3.77 deg
   Bank =  -0.02 deg
   Pitch =   3.77 deg
```

```
   Heading =   0.02 deg
   Altitude =   200.00 m

Extracting aircraft linear model...


 Longitudinal Dynamics
-----------------------
  State vector: x = [u w q theta h Omega]
  Input vector: u = [elevator throttle]
 Output vector: y = [Va alpha q theta h]
 State matrix: A =
   -0.2197    0.6002   -1.4882   -9.7967   -0.0001    0.0108
   -0.5820   -4.1204   22.4024   -0.6461    0.0009         0
    0.4823   -4.5284   -4.7512         0    0.0000   -0.0084
         0         0    1.0000         0         0         0
    0.0658   -0.9978         0   22.9997         0         0
   32.1012    2.1170         0         0   -0.0295   -2.7813


 Control matrix: B =
    0.3246         0
   -2.1520         0
  -29.8216         0
         0         0
         0         0
         0  448.5357


 Observation matrix: C =
    0.9978    0.0658         0         0         0         0
   -0.0029    0.0434         0         0         0         0
         0         0    1.0000         0         0         0
         0         0         0    1.0000         0         0
         0         0         0         0    1.0000         0


 Eigenvalue: -4.4337 +/- 10.1009 i
 Damping = 0.4019, natural frequency = 11.0311 rad/s, period =   0.6220 s
 Eigenvalue: -2.8950
 Time constant = 0.3454 s
 Eigenvalue: -0.0549 +/- 0.5690 i
 Damping = 0.0961, natural frequency = 0.5716 rad/s, period =  11.0431 s
 Eigenvalue: -0.0005
 Time constant = 1931.1123 s


 Lateral-directional Dynamics
------------------------------
  State vector: x = [v p r phi psi]
  Input vector: u = [aileron rudder]
 Output vector: y = [beta p r phi psi]
 State matrix: A =
   -0.6373    1.5135  -22.9498    9.7967         0
   -4.1919  -20.6283    9.9282         0         0
    0.6798   -2.6757   -1.0377         0         0
         0    1.0000    0.0659   -0.0000         0
         0         0    1.0022   -0.0000         0


 Control matrix: B =
   -1.2510    3.1931
 -109.8373    1.9763
   -4.3307  -20.1754
         0         0
         0         0


 Observation matrix: C =
    0.0435         0         0         0         0
         0    1.0000         0         0         0
         0         0    1.0000         0         0
         0         0         0    1.0000         0
         0         0         0         0    1.0000


 Eigenvalue: -19.7263
 Time constant = 0.0507 s
 Eigenvalue: -1.3190 +/- 5.5958 i
 Damping = 0.2294, natural frequency = 5.7492 rad/s, period =   1.1228 s
 Eigenvalue: 0.0611
 Time constant = -16.3757 s
```

3 Setting-up and Running an Aircraft Model

This chapter outlines the pre-built aircraft models that are provided in the **AeroSim** blockset. These can be customized by setting the aircraft aerodynamic, propulsion, and inertia properties in an aircraft parameter file. The general architecture of the complete aircraft model blocks is also presented. Finally, special issues such as interfacing the Simulink models with Flight Simulator are discussed.

3.1 Aircraft Model Examples

The following aircraft examples are included in the **AeroSim** blockset:

- Aerosonde UAV - a small autonomous airplane designed for weather-reconnaissance and remote-sensing missions - Fig. 25. The Aerosonde block can be found in the demo models

Figure 25: Aerosonde UAV

aerosonde_demo1 through **6**. It is using the **6-DOF Aircraft Model (with body-axes EOM)** block from the **AeroSim** library. The aircraft parameter file is **aerosondecfg.mat** which was generated by the **aerosondeconfig.m** script.

- Navion - a high-performance general-aviation airplane - Fig. 26 The Navion block can be found in the demo models **navion_demo1**

Figure 26: Navion

through **5**. It is also using the **6-DOF Aircraft Model (with body-axes EOM)** block from the **AeroSim** library. The aircraft parameter file is **navioncfg.mat** which was generated by the **navionconfig.m** script.

3.2 Building an aircraft configuration

A new aircraft parameter file can be generated from a custom Matlab script. To create this script, you would need to open the template aircraft configuration script **config_template.m** which can be found in the **samples** directory, and specify the aircraft aerodynamic, propulsion and inertia parameters.

By running this script at the Matlab command prompt, a new aircraft parameter file of the form **filename.mat** will be created. The file name can then be used in any of the complete aircraft blocks available in the **AeroSim** library.

The first variable that should be specified is the name of the aircraft parameter file that will be generated. Type the chosen name of your aircraft parameter file, as a string, without the .mat extension:

```
% Insert the name of the MAT-file that will be generated
% (without .mat extension)
cfgmatfile = 'myairplanecfg';
```

3.2.1 Conventions

Fig. 27 shows the various reference points required throughout the aircraft configuration script. The locations of these points will have to be specified with respect to the arbitrarily-chosen origin of the body-frame. These reference points include the CG locations for zero-fuel and for full-tank, the application points for the aerodynamic and propulsion forces.

The orientation of the body axes should be chosen as in the Fig. 27 - X should point forward, Y toward the right wing tip, and Z down.

For consistency, we recommend that all parameters be given in metric (SI) units. Conversion to English units can be performed in the Simulink model, by using the unit conversion blocks provided with the **AeroSim** library.

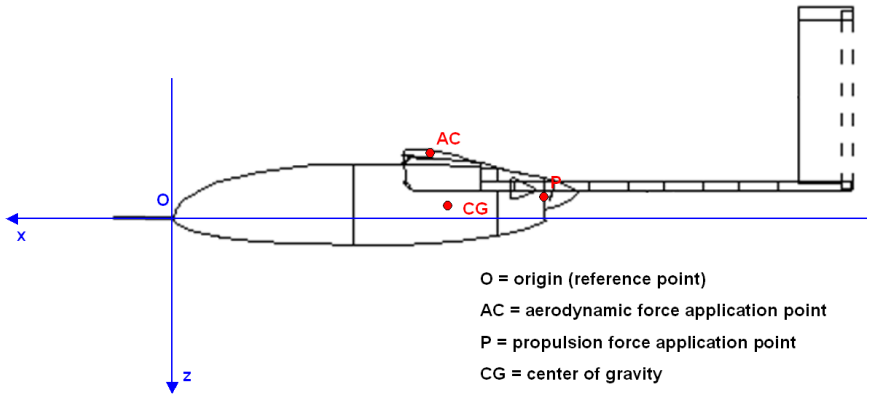


Figure 27: Airframe reference points

3.2.2 Section 1: Aerodynamics

Section 1 of the aircraft configuration script specifies the aerodynamic parameters of the aircraft. These are explained next:

- Reference point = specify the location, with respect to the body-frame origin, of the aerodynamic forces application point. This should be a 1x3 row vector with the x, y, and z coordinates. All the aerodynamic coefficients (force and moment coefficients) should be given referenced to this point.
- Aerodynamic parameter bounds = the limits that the aircraft model will impose on the airspeed, sideslip, and angle-of-attack, given as 1x2 vector of min and max values. The pur-

pose of using these limits is to keep the outputs of the aerodynamic model within the linear region. The component build-up method implemented by the aerodynamic model in the **AeroSim** library uses first order terms, therefore the aerodynamic model will provide acceptable results only in the linear aerodynamics conditions (small angles).

- Aerodynamic reference parameters = the reference parameters for which the aerodynamic coefficients were normalized. These include the reference wing chord, the wing span, and the reference wing area.

- Lift coefficient terms. The lift coefficient is computed using the expression (1).

$$C_L = C_{L0} + C_L^{\alpha} \cdot \alpha + C_L^{\delta_f} \cdot \delta_f + C_L^{\delta_e} \cdot \delta_e + \\ + \frac{c}{2V_a} \left(C_L^{\dot{\alpha}} \cdot \dot{\alpha} + C_L^{q} \cdot q \right) + C_L^{M} \cdot M \tag{1}$$

- Drag coefficient terms. The drag coefficient is computed using the expression (2).

$$C_D = C_{D0} + \frac{(C_L - C_{L0})^2}{\pi e AR} + C_D^{\delta_f} \cdot \delta_f + C_D^{\delta_e} \cdot \delta_e + \\ + C_D^{\delta_a} \cdot \delta_a + C_D^{\delta_r} \cdot \delta_r + C_L^{M} \cdot M \tag{2}$$

- Side force coefficient terms. The side force coefficient is computed using the expression (3).

$$C_Y = C_Y^{\beta} \cdot \beta + C_Y^{\delta_a} \cdot \delta_a + C_Y^{\delta_r} \cdot \delta_r + \\ + \frac{b}{2V_a} \left(C_Y^{p} \cdot p + C_Y^{r} \cdot r \right) \tag{3}$$

- Pitch moment coefficient terms. The pitch moment coefficient is computed using the expression (4).

$$C_m = C_{m0} + C_m^{\alpha} \cdot \alpha + C_m^{\delta_f} \cdot \delta_f + C_m^{\delta_e} \cdot \delta_e + \\ + \frac{c}{2V_a} \left(C_m^{\dot{\alpha}} \cdot \dot{\alpha} + C_m^{q} \cdot q \right) + C_m^{M} \cdot M \tag{4}$$

- Roll moment coefficient terms. The roll moment coefficient is computed using the expression (5).

$$C_l = C_l^{\beta} \cdot \beta + C_l^{\delta_a} \cdot \delta_a + C_l^{\delta_r} \cdot \delta_r + \\ + \frac{b}{2V_a} \left(C_l^{p} \cdot p + C_l^{r} \cdot r \right) \tag{5}$$

- Yaw moment coefficient terms. The yaw moment coefficient is computed using the expression (6).

$$C_n = C_n^{\beta} \cdot \beta + C_n^{\delta_a} \cdot \delta_a + C_n^{\delta_r} \cdot \delta_r + \\ + \frac{b}{2V_a} \left(C_n^{p} \cdot p + C_n^{r} \cdot r \right) \tag{6}$$

3.2.3 Section 2: Propeller

The second section of the aircraft configuration script specifies the geometry and aerodynamic performance of the propeller.

- Propeller hub location = the position of the propulsion force and moment application point, given with respect to the body-frame origin. The location is specified as a 1x3 row vector of x, y, and z coordinates.

- Advance ratio. The aerodynamic performance of the propeller should be given as a look-up table of propeller coefficients (CP and CT) as functions of the propeller advance ratio. This variable specifies the advance ratio vector which corresponds to the look-up table.

- Coefficient of thrust. The vector of coefficients of thrust for the advance ratios given above (the vector should have the same size).

- Coefficient of power. The vector of coefficients of power for the advance ratios given above (the vector should have the same size).

- Propeller radius = the radius of the propeller is used by the propulsion model to compute the force and torque from the normalized coefficients. These loads are computed as follows:

$$F_p = \frac{4}{\pi^2}\rho R^4 \Omega^2 C_T \tag{7}$$

$$M_p = -\frac{4}{\pi^3}\rho R^5 \Omega^2 C_P \tag{8}$$

- Propeller inertia = the propeller moment of inertia is used by the propulsion equation of motion (dynamics) to solve for the current rotation speed.

3.2.4 Section 3: Engine

The third section of the aircraft configuration scripts allows the user to specify the engine characteristics. All engine data is given at sea-level. The engine model will correct the data for altitude effects. For a normally-aspirated general aviation piston engine, this includes the following parameters:

- RPM = the vector of engine speeds for which the engine data is given, in rotations-per-minute. All engine parameters are specified as 2-D look-up tables (functions of engine speed and intake manifold pressure).

- MAP = the vector of manifold pressures for which the engine data is given, in kilo-pascals.

- Fuel flow = the sea-level fuel flow as a function of RPM and MAP. The number of rows in the matrix should match the size of the RPM vector, the number of columns should match the size of the MAP vector.

- Power = the engine power at sea-level, as a function of RPM and MAP. The number of rows in the matrix should match the size of the RPM vector, the number of columns should match the size of the MAP vector.

- Sea-level atmospheric conditions = the sea-level atmospheric conditions, including pressure in Pascals and temperature in degrees Kelvin, for which the engine data above is given.

- Engine shaft inertia = the moment of inertia of the rotating parts of the engine. This is added to the propeller inertia and used in the propulsion equation of motion to compute the current engine speed. Generally, the engine shaft inertia is significantly lower than that of the propeller, and it can be neglected without any major effects over the aircraft dynamics.

3.2.5 Section 4: Inertia

The fourth section of the aircraft configuration script specifies the inertia parameters of the aircraft: mass, CG location, and moments of inertia.

- Empty aircraft mass = the mass of the aircraft without fuel.
- Gross aircraft mass = the mass of the aircraft with the fuel tank full.
- CG empty = the position of the center of gravity for the aircraft without fuel. This is provided as a 1x3 vector of x, y, z coordinates with respect to the origin.
- CG gross = the position of the center of gravity for the aircraft with full fuel tank. This is provided as a 1x3 vector of x, y, z coordinates with respect to the origin.
- Empty moments of inertia = the moments of inertia of the aircraft without fuel. These are provided as a 1x4 vector of moment of inertia about body axes - J_x, J_y, J_z, and J_{xz}. For most aircraft the inertia cross-products J_{xy} and J_{yz} can be neglected because of the symmetry about the x-z plane.
- Gross moments of inertia = the moments of inertia of the aircraft with full fuel tank. These are provided as a 1x4 vector of moment of inertia about body axes - J_x, J_y, J_z, and J_{xz}.

3.2.6 Section 5: Other parameters

In the last section of the aircraft configuration script, the user can set other, less important parameters. At this point, the only variable that can be edited is the calendar date used by the World Magnetic Model to compute the magnetic field at aircraft location. The date is specified as a 1x3 vector of day, month, and year.

3.3 The pre-built Aircraft Models

Once the editing of the aircraft configuration script is completed, it should be saved with a unique name, for example **myairplaneconfig.m**.

By executing the script **myairplaneconfig.m** at the Matlab command prompt, an aircraft configuration file - **myairplanecfg.mat** will be created in the present working directory.

We can then start a new Simulink model, and add one of the complete aircraft blocks from the **AeroSim** library, such as the one shown in Fig. 28.

Next we double-click the block to open the block parameters dialog. Here we will specify the aircraft parameter file, **myairplane.mat**, the initial conditions (position, velocity, attitude, angular rates, fuel, engine speed), the ground altitude with respect to the sea-level, and the simulation sample time.

Finally, go to the Simulink model pull-down menu, under Simulation and open the **Simulation parameters**. Set the solver type to **Fixed-step**, the integration scheme to **ode4** or **ode5**, and set the fixed step size to match the aircraft model sample time.

Before starting the simulation, we can also add a constant source for the Controls input to the aircraft model. This will provide the model with a set of constant actuator commands. If we decide to leave the Controls unconnected, the aircraft will fly with the engine off and control surfaces at neutral positions.

The internal layout of the complete aircraft model is shown in Fig. 29. All sub-systems were designed using the **AeroSim** blocks.

A simplified diagram is presented in Fig. 30. The aerodynamics, propulsion, and inertia models compute the airframe loads

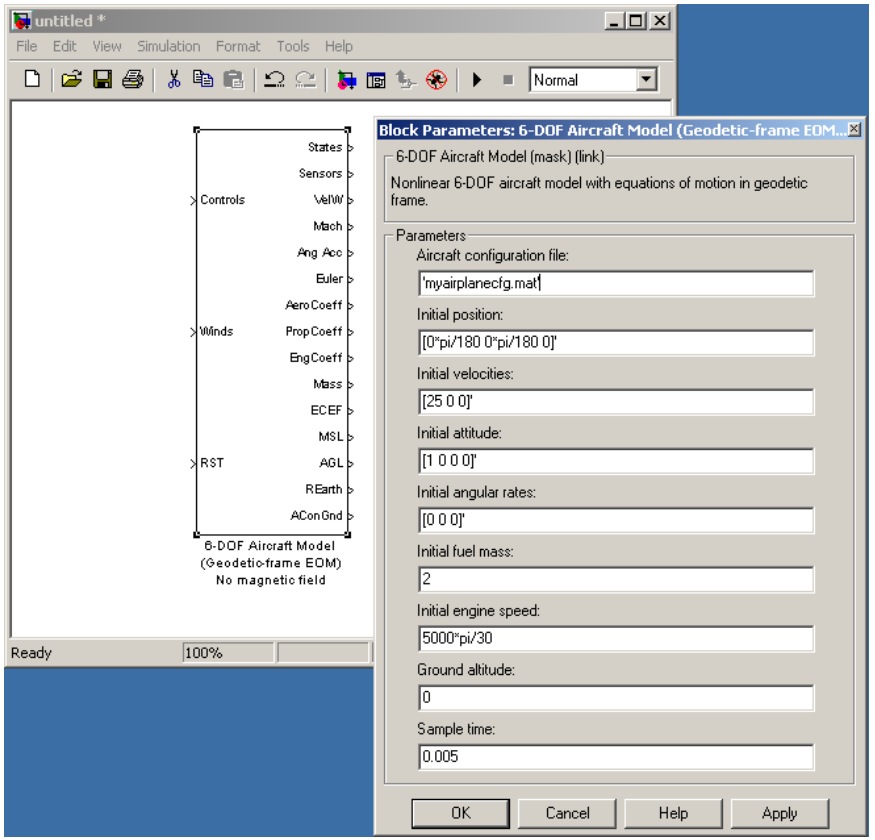


Figure 28: Complete aircraft block and options dialog

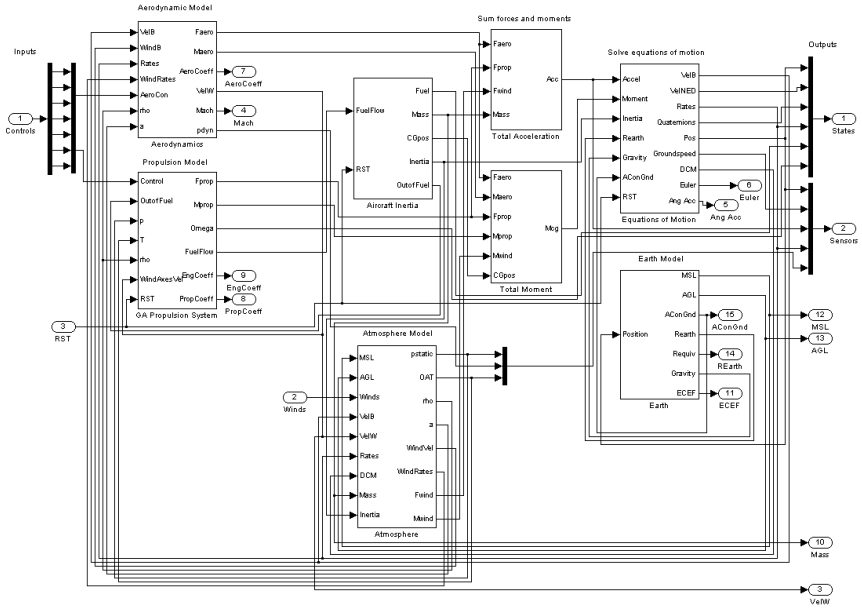


Figure 29: Internal structure of the complete aircraft model

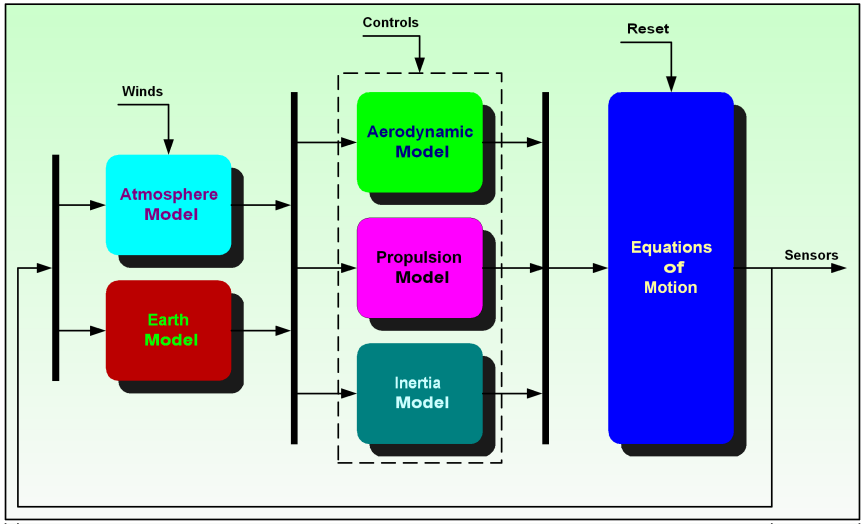


Figure 30: Simplified internal structure diagram

(forces and moments) as functions of control inputs and environment (atmosphere and Earth) effects. The resulting accelerations are then integrated by the Equations of Motion to obtain the aircraft states (position, velocity, attitude, angular velocities). The aircraft states then will affect the output of the environment blocks at the next iteration (for example altitude changes result in atmospheric pressure changes, latitude and longitude variations result in gravity variations). Also, the aircraft states are used in the computation of sensor outputs (GPS, inertial measurements, etc.).

3.4 Using FlightGear Aircraft Configuration Files

FlightGear is an open-source multi-platform cooperative flight simulator project. Source code for the entire project is available and licensed under the GNU General Public License. The goal of the FlightGear project is to create a sophisticated flight simulator framework for use in research or academic environments, for the development and pursuit of other interesting flight simulation ideas, and as an end-user application. For more information please visit the main FlightGear website, at
http://www.flightgear.org.

Initially, the FlightGear project used the **LaRCSim** flight dynamics code for the simulation of aircraft dynamics. However, the LaRCSim software was inflexible in defining the aircraft parameters (any change to the default aircraft - Navion - parameters required manual editing of the LaRCSim source code). Due to this fact, several new flight dynamic codes for FlightGear were created and are in various stages of development. Of these, the best known project is **JSBSim**, an open-source flight dynamics model which can be used either from FlightGear or independent of the visual interface. For more information about the JSBSim project, visit the main JSBSim website, at
http://jsbsim.sourceforge.net.

3.4.1 The JSBSim XML Configuration File

The main strength of JSBSim is that it provides flexibility to the flight dynamics developer by accepting aircraft configurations as XML files which are structured and easy to read. The lack of standards for exchange of aircraft models in the flight simulation community is well known and poses problems since most aircraft development projects involve several contractors and these would have to exchange flight dynamic models on a regular basis. The open-source community, and JSBSim in particular, have taken the first steps in establishing an XML-based standard for aircraft models.

A detailed and evolving document that describes the JSBSim configuration or definition files can be found on the main JSBSim website. A brief presentation of these files is done next.

There are three types of JSBSim definition files:

1. Aircraft Definition File, which describes the airframe geometry and inertia parameters, the aerodynamics, the flight control system, and which types of engine(s) and thruster(s) the aircraft has.

2. Engine Definition File, which describes the properties of a specific engine. At this time JSBSim supports piston and rocket engines, but turbine engine models are under development.

3. Thruster Definition File, which describes the properties of a specific thruster. At this time JSBSim provides models for propellers and nozzles.

This layout permits development of aircraft models in a structured fashion in which the same aircraft can be easily fitted with a different engine or propeller. The engine and thruster components can be reused for new aircraft configurations.

The Aircraft Definition File is structured in the following subsections, defined as xml tags:

- *METRICS*, which describes the aircraft geometry and inertia parameters
- *UNDERCARRIAGE*, which describes the landing gear and ground contact points,
- *PROPULSION*, which specifies which engine(s) and thruster(s) are used, what is their location in the aircraft coordinates. It also specifies the fuel/oxydizer tanks.
- *FLIGHT_CONTROL*, which specifies the aircraft flight control laws.
- *AERODYNAMICS*, which includes all of the aerodynamic coefficients and scaling parameters required for computing the aerodynamic forces and moments.
- *OUTPUT*, which specifies what parameters of the flight dynamics model the program will output.

3.4.2 The xmlAircraft Parser

The XML aircraft parsing utilities are located in the **fgutil** directory. The system is based on the *xml_parser*, a generic XML parser utility for Matlab, written by Guillaume Flandin. The generic XML parser is used to load any xml file into a Matlab xml tree structure. The AeroSim Blockset adds the following aircraft-specific parsing scripts:

- *xmlEngine*, which can parse an engine definition file into a Matlab engine structure.
- *xmlThruster*, which can parse a thruster definition file into a Matlab thruster structure.
- *xmlSearchForTag*, which searches an XML tree for specific tags and returns their UIDs.
- *xmlParameter*, which searches a character string for specific parameter strings and returns their corresponding data (scalar or vector).
- *xmlMetrics*, which parses the METRICS tag of an aircraft XML tree into a Matlab metrics structure.
- *xmlTank*, which parses a propellant tank tag into a Matlab tank structure.
- *xmlPropulsion*, which parses the PROPULSION tag of an aircraft XML tree into a Matlab propulsion structure.
- *xmlAeroCoeff*, which parses an aerodynamic coefficient tag of an aircraft XML tree into a Matlab coefficient structure.
- *xmlAeroFactor*, which parses an aerodynamic factor tag of on aircraft XML tree into a Matlab factor structure.
- *xmlAeroGroup*, which parses an aerodynamic factor group tag into a Matlab factor group structure.
- *xmlAeroAxis*, which parses an aerodynamic axis tag into a Matlab aerodynamic axis structure.
- *xmlAerodynamics*, which parses the AERODYNAMICS tag of an aircraft XML tree into a Matlab aerodynamics structure.

- *xmlAircraft*, which is the main aircraft definition file parser. The function reads an aircraft definition file along with its engine and thruster definition files and parses the complete aircraft model into a Matlab aircraft structure. The aircraft structure is documented in the next subsection.

- *AircraftUnitConv*, which is an unit conversion function that processes a Matlab aircraft structure returned by *xmlAircraft* (JSBSim uses English units) and converts all parameters to metric units.

3.4.3 The Matlab Aircraft Structure

To parse an aircraft, we will use the *xmlAircraft* utility. For example, if the path to FlightGear is **c:flightgear-0.8** then we would use the following Matlab command:

```
c172 = xmlAircraft('c172', 'c:\flightgear-0.8')
```

The newly created Matlab aircraft structure is *c172* and this contains:

```
c172 =

           Name: 'c172'
        Version: '1.58'
        Metrics: [1x1 struct]
     Propulsion: [1x1 struct]
  Aerodynamics: [1x1 struct]
```

Besides the aircraft name and version strings, the aircraft structure contains other structures such as Metrics, Propulsion, and Aerodynamics.

We can now list the metrics structure by typing *c172.Metrics* at the Matlab prompt, to get:

```
>> c172.Metrics

ans =

       WingArea: 174
       WingSpan: 35.8000
      WingChord: 4.9000
      HTailArea: 21.9000
       HTailArm: 15.7000
      VTailArea: 16.5000
       VTailArm: 15.7000
       InertiaX: 948
       InertiaY: 1346
       InertiaZ: 1967
      InertiaXZ: 0
    EmptyWeight: 1500
          ACLoc: [3x1 double]
          CGLoc: [3x1 double]
     PointMass: {[4x1 double]}
      PilotLoc: [3x1 double]
```

Individual aircraft parameters can be retrieved in a similar fashion. For example, if we want to see the aircraft CG location, we could type *c172.Metrics.CGLoc* to get:

```
>> c172.Metrics.CGLoc

ans =

   41.0000
         0
   36.5000
```

The PointMass variable can contain more than one point mass (for example pilot + passengers), so it is saved as a structure of arrays. To see the data for one of the point-masses, we can refer to the structure index - for example, we can type *c172.Metrics.PointMass*{*1*} to get:

```
>> c172.Metrics.PointMass{1}
```

```
ans =

   180
    36
   -14
    24
```

which represents the mass and the location in aircraft coordinates of the first point-mass.

To take a quick look at the propulsion model data, type *c172.Propulsion*. The output is shown below:

```
>> c172.Propulsion

ans =

      Engine: {[1x1 struct]}
    Thruster: {[1x1 struct]}
        Tank: {[1x1 struct]  [1x1 struct]}
```

Since the aircraft can have one or more engines, thrusters, respectively propellant tanks, they are loaded as structures. To get the structure of one of the engines we could again use the index, by typing *c172.Propulsion.Engine*{*1*} to get:

```
>> c172.Propulsion.Engine{1}

ans =

     Name: 'IO360C'
     Type: 'PISTON'
    Param: [1x1 struct]
      Loc: [-19.7000 0 26.6000]
    Pitch: 0
      Yaw: 0
     Feed: [0 1]
```

We can get one more level deeper by taking a look at the specific engine parameters:

```
>> c172.Propulsion.Engine{1}.Param

ans =

          MAPLim: [6.5000 28.5000]
    Displacement: 360
        MaxPower: 160
         NCycles: 2
         IdleRPM: 900
     ThrottleLim: [0.2000 1]
```

Similarly, the data for the thruster and fuel tank can be accessed.

If we take a look at the aerodynamics sub-structure we find:

```
>> c172.Aerodynamics

ans =

    AlphaLim: [2x1 double]
     HystLim: [2x1 double]
        Lift: [1x1 struct]
        Drag: [1x1 struct]
        Side: [1x1 struct]
        Roll: [1x1 struct]
       Pitch: [1x1 struct]
         Yaw: [1x1 struct]
```

Each of the 6 aerodynamic axes (lift, drag, side, roll, pitch, yaw) contains aerodynamic groups and coefficients. For example, the lift axis contains one factor group (Basic lift) and 3 additional coefficients:

```
>> c172.Aerodynamics.Lift

ans =

      Name: 'LIFT'
     Basic: [1x1 struct]
      CLDe: [1x1 struct]
    CLadot: [1x1 struct]
       CLq: [1x1 struct]
```

Inside the basic lift group we will find the group factor which augments all of the group coefficients for ground effect, and the 2 main lift coefficients - the AOA dependence and the flap deflection effect.

```
>> c172.Aerodynamics.Lift.Basic

ans =

           Name: 'CLb'
    Description: 'Basic_lift'
          kCLge: [1x1 struct]
          CLwbh: [1x1 struct]
           CLDf: [1x1 struct]
```

If we go one level deeper to take a closer look at the lift coefficient due to angle-of-attack we have:

```
>> c172.Aerodynamics.Lift.Basic.CLwbh

ans =

           Name: 'CLwbh'
           Type: 'TABLE'
    Description: 'Lift_due_to_alpha'
       RowIndex: 'FG_ALPHA'
       ColIndex: 'FG_HYSTPARM'
         ColArg: [2x1 double]
         RowArg: [17x1 double]
           Data: [17x2 double]
```

We can easily see that this coefficient is a table. Recall that aerodynamic coefficients in JSBSim can be defined as either Values, Vectors, or Tables.

The row argument *RowArg* is the angle of attack *alpha* while the column argument *ColArg* is the hysteresis parameter (the lift coefficient hysteresis appears when recovering the aircraft from a stall).

The actual data for this coefficient can be retrieved:

```
>> c172.Aerodynamics.Lift.Basic.CLwbh.Data

ans =

   -0.2200   -0.2200
    0.2500    0.2500
    0.7300    0.7300
    0.8300    0.7800
    0.9200    0.7900
    1.0200    0.8100
    1.0800    0.8200
    1.1300    0.8300
    1.1900    0.8500
    1.2500    0.8600
    1.3500    0.8800
    1.4400    0.9000
    1.4700    0.9200
    1.4300    0.9500
    1.3800    0.9900
    1.3000    1.0500
    1.1500    1.1500
```

3.5 Additional Matlab Utilities

The directory **util** contains two small Matlab functions.

- *eigparam* computes the damping, period, natural and damped frequencies of a complex eigenvalue. The function is used by the aircraft trim and linearization scripts to display the characteristics of the linear aircraft models.
- *euler2quat* computes the quaternion vector corresponding to the Euler angle representation. The function can be used to compute the initial quaternions required for an aircraft model based on the initial aircraft attitude expressed in Euler angles.

4 Block Reference

The AeroSim library includes all of the blocks needed for building a nonlinear six-degree-of-freedom aircraft model. This section provides a detailed description for each block that can be found in the library. In addition to the information in the User's Guide, each block is provided with online help for quick reference. To access it, right-click on any block and select **Help** from the pop-up menu.

The main library folder, shown in Fig. 31 includes sub-folders for various parts of the aircraft dynamic model. The sub-sections of the **Block Reference** section correspond to these library sub-folders. The **AeroSim** library contains a total of 103 blocks. Almost all of them are implemented using basic Simulink blocks, with the exception of a few such as the WMM-2000 Earth magnetic model, and the pilot interface blocks which use operating system calls unavailable in Simulink - these are implemented using C/C++ as C-MEX S-functions. The source code for these S-function is provided with the **AeroSim** library.

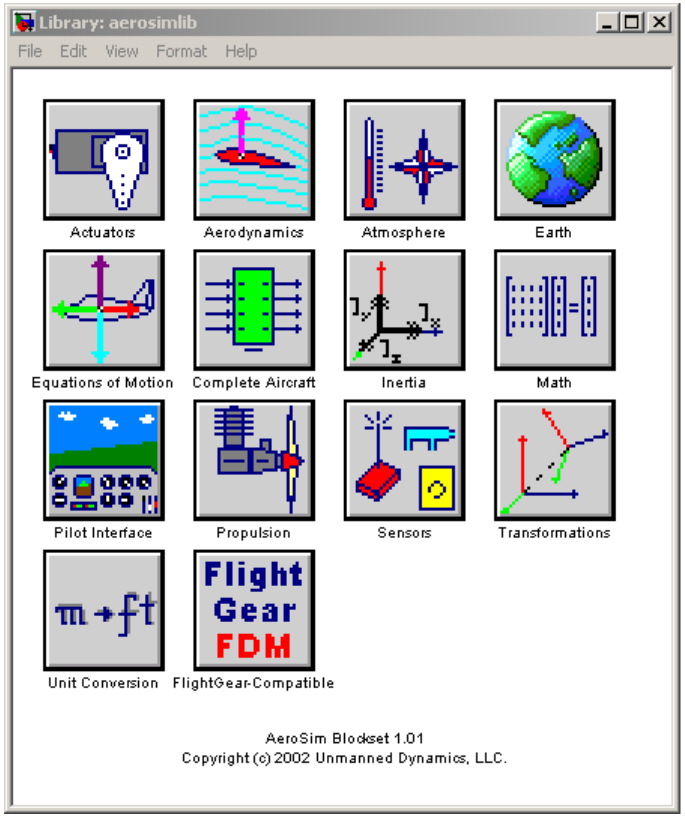


Figure 31: AeroSim Library

4.1 Actuators

The **Actuators** folder contains generic models of electro-mechanical actuators used in flight control systems. Although a fully-functional aircraft dynamic model can be created without making use of these blocks, the actuator models are very useful for testing the stability and performance of airplane-autopilot closed-loop systems.

4.1.1 Simple Actuator (1st-order dynamics)

The block provides a simple actuator model featuring signal scaling, 1-st order dynamics (time lag), saturation and rate limits. The block representation is shown in Fig. 32.

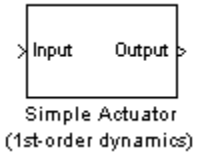


Figure 32: Simple Actuator Block

Block characteristics:

1. *Parameters:*
   - Max actuation rate = the maximum speed at which the actuator can move, in any direction. The speed must be expressed in actuator **output** units per unit time.
   - Input signal range = a 2×1 row vector that contains the minimum and the maximum values of the input signal, expressed in input signal units.
   - Actuator range = a 2×1 row vector that contains the minimum and the maximum values of the actuator output (deflection), expressed in output signal (deflection) units.
   - Actuator lag = the time constant τ_a of the actuator linear dynamics which are implemented as a first-order transfer function (9).

$$G_a(s) = \frac{1}{\tau_a s + 1} \tag{9}$$

2. *Inputs:*
   - Input = the input signal (command), expressed in input signal units.
3. *Outputs:*
   - Output = the output signal (deflection), expressed in output signal units.
4. *Details:* As an example, consider a simple actuator such as an R/C model airplane PWM servo connected to elevator. The servo receives its pulse-width command from a digital flight control system as an integer count, where 0 counts will position the elevator to a minimum deflection of -30° and 1024 counts will send the elevator to the maximum deflection of $+30^\circ$. Assume that the maximum actuation rate is $45^\circ/s$, based on servo performance and servo-elevator gear ratio. This actuator can be easily modelled using this block, by choosing the actuator rate as $45^\circ/s$, the input signal range as $[0 \quad 1024]$, the actuator range as $[-30 \quad 30]$ and a 25 ms lag. The input to this signal will be provided in counts and the output will consist of an elevator deflection in degrees.
5. *Usage:* The actuator block gets its input from the user (pilot), or from an automatic flight control system. The actuator output signal should be connected to aerodynamic and/or

propulsion blocks that compute the forces and moments applied to the airframe due to the actuator deflection.

4.1.2 Simple Actuator (2nd-order dynamics)

The block provides a simple actuator model featuring signal scaling, 2-nd order dynamics, saturation and rate limits. The block representation is shown in Fig. 33.

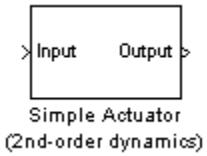


Figure 33: Simple Actuator Block

Block characteristics:

1. *Parameters:*
   - Max actuation rate = the maximum speed at which the actuator can move, in any direction. The speed must be expressed in actuator **output** units per unit time.
   - Input signal range = a 2×1 row vector that contains the minimum and the maximum values of the input signal, expressed in input signal units.
   - Actuator range = a 2×1 row vector that contains the minimum and the maximum values of the actuator output (deflection), expressed in output signal (deflection) units.
   - Bandwidth = the actuator bandwidth b at a gain $g = -3dB$ and phase $\phi = -90°$. The 2-nd order transfer function is implemented as in (10).

$$G_a(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{10}$$

   where $\omega_n = 2\pi b$ and $\zeta = \frac{1}{2}10^{\frac{-g}{20}}$.

2. *Inputs:*
   - Input = the input signal (command), expressed in input signal units.
3. *Outputs:*
   - Output = the output signal (deflection), expressed in output signal units.
4. *Details:* The functionality of this block is similar to that of the Simple Actuator with 1st-order dynamics (previous subsection).
5. *Usage:* The actuator block gets its input from the user (pilot), or from an automatic flight control system. The actuator output signal should be connected to aerodynamic and/or propulsion blocks that compute the forces and moments applied to the airframe due to the actuator deflection.

4.1.3 D/A Converter

The *D/A Converter* block converts the digital signal with a specified resolution to an analog voltage within a specified voltage range. The block is pictured in Fig. 34.

Figure 34: D/A Converter Block

Block characteristics:

1. *Parameters:*
   - Voltage range = a 1×2 vector containing the minimum and maximum values of the output voltage.
   - Resolution = the number of bits of resolution, which determine the number of counts available on the specified voltage range. For example, a 10-bit converter will provide $2^{10} = 1024$ counts.
   - Sample time = the time interval at which the digital signal is provided.
2. *Inputs:*
   - Digital = the digital signal, in counts.
3. *Outputs:*
   - Analog = the analog input signal, in volts.
4. *Details:* The block includes scale-factor conversion from counts to volts, saturation limits on the output signal.
5. *Usage:* The block can be used to convert actuator commands provided by a digital flight control system to analog actuator deflections.

4.2 Aerodynamics

The **Aerodynamics** library folder contains all the blocks required to develop a simple aerodynamic model of the aircraft. The approach taken in the **AeroSim** library involves linear aerodynamics in which the aerodynamic force and moment coefficients are computed using linear combinations of aerodynamic derivatives.

4.2.1 Aerodynamic Force

The aerodynamic force block, shown in Fig. 35, computes the aerodynamic force that acts on the airframe, based on the current aerodynamic force coefficients and dynamic pressure.

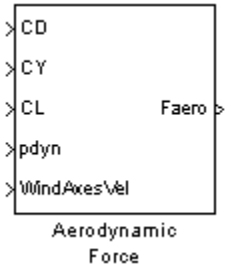


Figure 35: Aerodynamic Force Block

Block characteristics:

1. *Parameters:* The only parameter required is the wing reference area with respect to which the aerodynamic force coefficients were computed.

2. *Inputs:*

   - CD = the drag coefficient.
   - CY = the side force coefficient.
   - CL = the lift coefficient.
   - pdyn = the dynamic pressure $q = \frac{1}{2}\rho V_a^2$.
   - WindAxesVel = a 3×1 vector containing the airspeed V_a, the angle-of-attack α in radians and the sideslip angle β in radians.

3. *Outputs:*

   - Faero = the 3×1 vector of aerodynamic forces in body axes.

4. *Details:* The block implements the standard expressions for aerodynamic forces in wind-axes, as shown in (11), (12), and (13).

$$D = \frac{1}{2}\rho V_a^2 S C_D \tag{11}$$

$$Y = \frac{1}{2}\rho V_a^2 S C_Y \tag{12}$$

$$L = \frac{1}{2}\rho V_a^2 S C_L \tag{13}$$

   The forces are then expressed into body-axes using the *Body-Wind DCM* block, provided in AeroSim library as well (see subsection *Transformations*).

5. *Usage:* The block should get its inputs from the other aerodynamic blocks, that compute aerodynamic coefficients, dynamic pressure, and wind-axes velocities. The output from the *Aerodynamic Force* block should go to the *Total Acceleration* and *Total Moment* blocks to be summed-up with the other forces applied to the airframe.

4.2.2 Aerodynamic Moment

The aerodynamic moment block, shown in Fig. 36, computes the aerodynamic moment that acts on the airframe, based on the current aerodynamic moment coefficients and dynamic pressure.

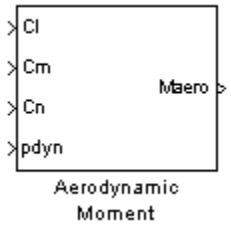


Figure 36: Aerodynamic Moment Block

Block characteristics:

1. *Parameters:*
   - Mean aerodynamic chord = the reference chord with respect to which the pitch moment coefficient was computed.
   - Wing span = the reference wing span with respect to which the roll and yaw moment coefficients were computed.
   - Wing reference area = the reference area with respect to which the moment coefficients were computed.
2. *Inputs:*
   - Cl = the current roll moment coefficient with respect to a reference point, such as the aerodynamic center of the wing.
   - Cm = the current pitch moment coefficient with respect to a reference point, such as the aerodynamic center of the wing.
   - Cn = the current yaw moment coefficient with respect to a reference point, such as the aerodynamic center of the wing.
   - pdyn = the dynamic pressure $q = \frac{1}{2}\rho V_a^2$.
3. *Outputs:*
   - Maero = the 3×1 vector of aerodynamic moments with respect to a reference point, such as the aerodynamic center of the wing.
4. *Details:* The block implements the standard expressions for aerodynamic moments, as shown in (14), (15), and (16).

$$L = \frac{1}{2}\rho V_a^2 S b C_l \tag{14}$$

$$M = \frac{1}{2}\rho V_a^2 S c C_m \tag{15}$$

$$N = \frac{1}{2}\rho V_a^2 S b C_n \tag{16}$$

5. *Usage:* The block should get its inputs from the other aerodynamic blocks, that compute aerodynamic coefficients and dynamic pressure. The output from the *Aerodynamic Moment* block should go to the *Total Moment* block to be summed-up with the other moments applied to the airframe.

4.2.3 Wind-axes Velocities

The block computes the wind-axes velocities based on groundspeed and wind speed. The name of "wind-axes velocities" is slightly misleading since the returned vector includes only one velocity - the airspeed - and two angles - angle-of-attack and sideslip. The block is represented in Fig. 37.

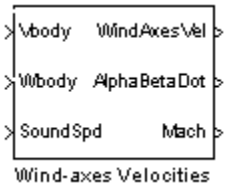


Figure 37: Wind-axes Velocity Block

Block characteristics:

1. *Parameters:*
   - Airspeed bounds = a 2×1 row vector containing the minimum and maximum allowed values for airspeed output.
   - Sideslip angle bounds = a 2×1 row vector containing the minimum and maximum allowed values for sideslip angle.
   - Angle-of-attack bounds = a 2×1 row vector containing the minimum and maximum allowed values for the angle-of-attack.
2. *Inputs:*
   - Vbody = a 3×1 vector containing the current groundspeed in body axes.
   - Wbody = a 3×1 vector containing the current wind speed in body axes.
   - SoundSpd = the speed of sound at current altitude.
3. *Outputs:*
   - WindAxesVel = a 3×1 vector of wind-axes velocities. The angle-of-attack and the sideslip are given in radians.
   - AlphaBetaDot = a 2×1 vector of angle-of-attack and sideslip time derivatives, in radians per second.
   - Mach = the current Mach number.
4. *Details:* The *Wind-axes Velocity* block computes first the airspeed in body axes $[u \quad v \quad w]$ by subtracting wind speed from groundspeed. The the airspeed magnitude V_a is computed using the *Vector Norm* block provided by AeroSim library. The angle-of-attack and sideslip are computed using the expressions (17) and (18).

$$\alpha = atan\left(\frac{w}{u}\right) \tag{17}$$

$$\beta = asin\left(\frac{v}{V_a}\right) \tag{18}$$

   Finally, the Mach number is computed as $M = \frac{V_a}{a}$.
5. *Usage:* The bounds specified in block parameters are chosen based on the envelope in which the aerodynamic coefficients

are expected to provide reasonably accurate aerodynamic behavior. For example, if the aerodynamic coefficients will be computed using the linear aerodynamic models provided in AeroSim library, then for a typical General Aviation airplane we would bound the angle-of attack to, let's say, -3° to 10°, and the side-slip angle to $\pm 5^\circ$. If the aerodynamic coefficients are provided as look-up tables, then we would bound the airflow angles such that table limits are not exceeded. The airspeed should be bounded according to the aircraft's flight envelope.

The groundspeed components in body axes are provided by the *Forces* block in *Equations of Motion*. The wind speed components are provided by the *Background Wind* and *Turbulence* blocks in *Atmosphere*. The speed of sound is provided by the *Standard Atmosphere* block in *Atmosphere*. The wind-axes velocities, the time derivatives of airflow angles, and the Mach number are used by many other *Aerodynamics* and *Propulsion* blocks.

4.2.4 Dynamic Pressure

The block computes the current dynamic pressure and it is represented in Fig. 38.

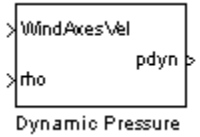


Figure 38: Dynamic Pressure Block

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - WindAxesVel = the 3×1 vector of wind-axes velocities (airspeed, angle-of-attack, and sideslip angle).
   - rho = air density at current altitude.
3. *Outputs:*
   - pdyn = current dynamic pressure.
4. *Details:* The *Dynamic Pressure* block has a fairly simple implementation, as shown in (19).

$$q = \frac{1}{2}\rho V a^2 \tag{19}$$

5. *Usage:* The *Dynamic Pressure* block gets its inputs from the *Wind-Axes Velocities* and *Standard Atmosphere* blocks. The dynamic pressure output is required in the computation of aerodynamic forces and moments. It can also be used as an aircraft model output since most aircraft are measuring it directly using the Pitot probe.

4.2.5 Lift Coefficient

The block computes the airplane non-dimensional lift coefficient (C_L) as a linear combination of individual contributions of various flight parameters. The block is represented in Fig. 39.

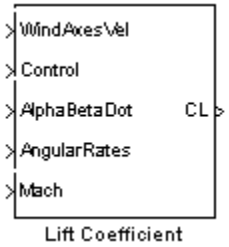


Figure 39: Lift Coefficient Block

Block characteristics:

1. *Parameters:*
   - Zero-alpha lift = the lift coefficient at zero angle-of-attack - C_{L0}.
   - alpha derivative = the variation (first-order derivative) of C_L with respect to the angle-of-attack - C_L^{α}.
   - Lift control derivative = the variation of C_L with lift control surface deflection (flap) - $C_L^{\delta_f}$.
   - Pitch control derivative = the variation of C_L with pitch control surface deflection (elevator) - $C_L^{\delta_e}$.
   - alpha_dot derivative = the variation of C_L with the time derivative of the angle-of-attack - $C_L^{\dot{\alpha}}$.
   - Pitch rate derivative = the variation of C_L with the pitch rate - C_L^{q}.
   - Mach number derivative = the variation of C_L with the Mach number - C_L^{M}.
   - Mean aerodynamic chord = the wing reference chord.
2. *Inputs:*
   - WindAxesVel = the 3×1 vector of wind-axes velocities $[V_a \quad \alpha \quad \beta]$.
   - Control = the 4×1 vector of aerodynamic controls (actuators) in the following order: lift control (flap), pitch control (elevator), roll control (aileron), and yaw control (rudder) $[\delta_f \quad \delta_e \quad \delta_a \quad \delta_r]$.
   - AlphaBetaDot = the 2×1 vector of time derivatives of airflow angles $[\dot{\alpha} \quad \dot{\beta}]$.
   - AngularRates = the 3×1 vector of body angular rates $[p \quad q \quad r]$.
   - Mach = the current Mach number.
3. *Outputs:*
   - CL = the current airplane lift coefficient.
4. *Details:* The lift coefficient is computed using the expression (20).

$$C_L = C_{L0} + C_L^{\alpha} \cdot \alpha + C_L^{\delta_f} \cdot \delta_f + C_L^{\delta_e} \cdot \delta_e + \\ + \frac{c}{2V_a} \left(C_L^{\dot{\alpha}} \cdot \dot{\alpha} + C_L^{q} \cdot q \right) + C_L^{M} \cdot M \tag{20}$$

5. *Usage:* The inputs should be provided by *Wind-axes Velocities* block in *Aerodynamics*, *Moments* block in *Equations of Motion*. Control input is provided by manual or automatic pilot. Block output is used by *Aerodynamic Force* block.

4.2.6 Drag Coefficient

The block computes the airplane non-dimensional drag coefficient (C_D) as a linear combination of individual contributions of various flight parameters. The block is represented in Fig. 40.

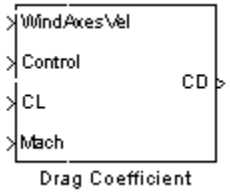


Figure 40: Drag Coefficient Block

Block characteristics:

1. *Parameters:*

   - Lift at minimum drag = the lift coefficient at minimum drag - C_{L0}.
   - Minimum drag = the minimum drag coefficient of the airplane C_{D0}.
   - Lift control derivative = the variation of C_D with lift control surface deflection (flap) - $C_D^{\delta_f}$.
   - Pitch control derivative = the variation of C_D with pitch control surface deflection (elevator) - $C_D^{\delta_e}$.
   - Roll control derivative = the variation of C_D with roll control surface deflection (aileron) - $C_D^{\delta_a}$.
   - Yaw control derivative = the variation of C_D with yaw control surface deflection (rudder) - $C_D^{\delta_r}$.
   - Mach number derivative = the variation of C_D with the Mach number - C_D^M.
   - Oswald's coefficient = the wing efficiency factor e, used for the parabolic approximation of the induced drag $C_{Di} = \frac{C_L^2}{\pi e AR}$.
   - Wing span = the reference wing span b.
   - Wing reference area = the wing reference area S which, together with the wing span, are used to compute the wing aspect ratio $AR = \frac{b^2}{S}$.

2. *Inputs:*

   - WindAxesVel = the 3×1 vector of wind-axes velocities $[V_a \quad \alpha \quad \beta]$.
   - Control = the 4×1 vector of aerodynamic controls (actuators) in the following order: lift control (flap), pitch control (elevator), roll control (aileron), and yaw control (rudder) $[\delta_f \quad \delta_e \quad \delta_a \quad \delta_r]$.
   - CL = the lift coefficient.
   - Mach = the current Mach number.

3. *Outputs:*

   - CD = the current airplane drag coefficient.

4. *Details:* The drag coefficient is computed using the expression (21).

$$C_D = C_{D0} + \frac{(C_L - C_{L0})^2}{\pi e AR} + C_D^{\delta_f} \cdot \delta_f + C_D^{\delta_e} \cdot \delta_e + \\ + C_D^{\delta_a} \cdot \delta_a + C_D^{\delta_r} \cdot \delta_r + C_L^M \cdot M \tag{21}$$

5. *Usage:* The inputs should be provided by *Wind-axes Velocities* and *Lift Coefficient* blocks in *Aerodynamics*. Control input is provided by manual or automatic pilot. Block output is used by *Aerodynamic Force* block.

4.2.7 Side Force coefficient

The block computes the airplane non-dimensional side force coefficient (C_Y) as a linear combination of individual contributions of various flight parameters. The block is represented in Fig. 41.

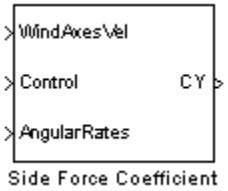


Figure 41: Side Force Coefficient Block

Block characteristics:

1. *Parameters:*
   - Sideslip derivative = the variation (first-order derivative) of C_Y with respect to the sideslip angle - C_Y^{β}.
   - Roll control derivative = the variation of C_Y with roll control surface deflection (aileron) - $C_Y^{\delta_a}$.
   - Yaw control derivative = the variation of C_Y with yaw control surface deflection (rudder) - $C_Y^{\delta_r}$.
   - Roll rate derivative = the variation of C_Y with the roll rate - C_Y^{p}.
   - Yaw rate derivative = the variation of C_Y with the yaw rate - C_Y^{r}.
   - Wing span = the reference wing span b.
2. *Inputs:*
   - WindAxesVel = the 3×1 vector of wind-axes velocities $[V_a \quad \alpha \quad \beta]$.
   - Control = the 4×1 vector of aerodynamic controls (actuators) in the following order: lift control (flap), pitch control (elevator), roll control (aileron), and yaw control (rudder) $[\delta_f \quad \delta_e \quad \delta_a \quad \delta_r]$.
   - AngularRates = the 3×1 vector of body angular rates $[p \quad q \quad r]$.
3. *Outputs:*
   - CY = the current airplane side force coefficient.
4. *Details:* The side force coefficient is computed using the expression (22).

$$C_Y = C_Y^{\beta} \cdot \beta + C_Y^{\delta_a} \cdot \delta_a + C_Y^{\delta_r} \cdot \delta_r + \\ + \frac{b}{2V_a} \left(C_Y^{p} \cdot p + C_Y^{r} \cdot r \right) \tag{22}$$

5. *Usage:* The inputs should be provided by *Wind-axes Velocities* block in *Aerodynamics*, *Moments* block in *Equations of Motion*. Control input is provided by manual or automatic pilot. Block output is used by *Aerodynamic Force* block.

4.2.8 Pitch Moment Coefficient

The block computes the airplane non-dimensional pitch moment coefficient (C_m) as a linear combination of individual contributions of various flight parameters. The block is represented in Fig. 42.

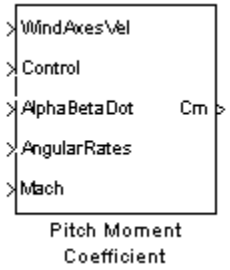


Figure 42: Pitch Moment Coefficient Block

Block characteristics:

1. *Parameters:*
   - Zero-alpha pitch = the pitch moment at zero angle-of-attack - C_{m0}.
   - alpha derivative = the variation (first-order derivative) of C_m with respect to the angle-of-attack - C_m^{α}.
   - Lift control derivative = the variation of C_m with lift control surface deflection (flap) - $C_m^{\delta_f}$.
   - Pitch control derivative = the variation of C_m with pitch control surface deflection (elevator) - $C_m^{\delta_e}$.
   - alpha_dot derivative = the variation of C_m with the time derivative of the angle-of-attack - $C_m^{\dot{\alpha}}$.
   - Pitch rate derivative = the variation of C_m with the pitch rate - C_m^q.
   - Mach number derivative = the variation of C_m with the Mach number - C_m^M.
   - Mean aerodynamic chord = the wing reference chord.
2. *Inputs:*
   - WindAxesVel = the 3×1 vector of wind-axes velocities $[V_a \quad \alpha \quad \beta]$.
   - Control = the 4×1 vector of aerodynamic controls (actuators) in the following order: lift control (flap), pitch control (elevator), roll control (aileron), and yaw control (rudder) $[\delta_f \quad \delta_e \quad \delta_a \quad \delta_r]$.
   - AlphaBetaDot = the 2×1 vector of time derivatives of airflow angles $[\dot{\alpha} \quad \dot{\beta}]$.
   - AngularRates = the 3×1 vector of body angular rates $[p \quad q \quad r]$.
   - Mach = the current Mach number.
3. *Outputs:*
   - Cm = the current airplane pitch moment coefficient.
4. *Details:* The pitch moment coefficient is computed using the expression (23).

$$C_m = C_{m0} + C_m^{\alpha} \cdot \alpha + C_m^{\delta_f} \cdot \delta_f + C_m^{\delta_e} \cdot \delta_e + \\ + \frac{c}{2V_a} \left(C_m^{\dot{\alpha}} \cdot \dot{\alpha} + C_m^q \cdot q \right) + C_m^M \cdot M \tag{23}$$

5. *Usage:* The inputs should be provided by *Wind-axes Velocities* block in *Aerodynamics*, *Moments* block in *Equations of Motion*. Control input is provided by manual or automatic pilot. Block output is used by *Aerodynamic Force* block.

4.2.9 Roll Moment Coefficient

The block computes the airplane non-dimensional roll moment coefficient (C_l) as a linear combination of individual contributions of various flight parameters. The block is represented in Fig. 43.

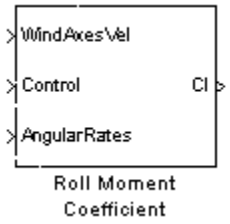


Figure 43: Roll Moment Coefficient Block

Block characteristics:

1. *Parameters:*
   - Sideslip derivative = the variation (first-order derivative) of C_l with respect to the sideslip angle - C_l^{β}.
   - Roll control derivative = the variation of C_l with roll control surface deflection (aileron) - $C_l^{\delta_a}$.
   - Yaw control derivative = the variation of C_l with yaw control surface deflection (rudder) - $C_l^{\delta_r}$.
   - Roll rate derivative = the variation of C_l with the roll rate - C_l^p.
   - Yaw rate derivative = the variation of C_l with the yaw rate - C_l^r.
   - Wing span = the reference wing span b.
2. *Inputs:*
   - WindAxesVel = the 3×1 vector of wind-axes velocities $[V_a \quad \alpha \quad \beta]$.
   - Control = the 4×1 vector of aerodynamic controls (actuators) in the following order: lift control (flap), pitch control (elevator), roll control (aileron), and yaw control (rudder) $[\delta_f \quad \delta_e \quad \delta_a \quad \delta_r]$.
   - AngularRates = the 3×1 vector of body angular rates $[p \quad q \quad r]$.
3. *Outputs:*
   - Cl = the current airplane roll moment coefficient.
4. *Details:* The roll moment coefficient is computed using the expression (24).

$$C_l = C_l^{\beta} \cdot \beta + C_l^{\delta_a} \cdot \delta_a + C_l^{\delta_r} \cdot \delta_r + \\ + \frac{b}{2V_a} \left(C_l^p \cdot p + C_l^r \cdot r \right) \tag{24}$$

5. *Usage:* The inputs should be provided by *Wind-axes Velocities* block in *Aerodynamics*, *Moments* block in *Equations of Motion*. Control input is provided by manual or automatic pilot. Block output is used by *Aerodynamic Force* block.

4.2.10 Yaw Moment Coefficient

The block computes the airplane non-dimensional yaw moment coefficient (C_n) as a linear combination of individual contributions of various flight parameters. The block is represented in Fig. 44.

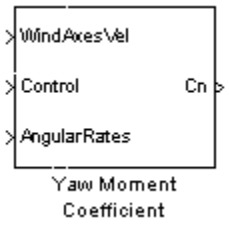


Figure 44: Yaw Moment Coefficient Block

Block characteristics:

1. *Parameters:*
   - Sideslip derivative = the variation (first-order derivative) of C_n with respect to the sideslip angle - C_n^{β}.
   - Roll control derivative = the variation of C_n with roll control surface deflection (aileron) - $C_n^{\delta_a}$.
   - Yaw control derivative = the variation of C_n with yaw control surface deflection (rudder) - $C_n^{\delta_r}$.
   - Roll rate derivative = the variation of C_n with the roll rate - C_n^{p}.
   - Yaw rate derivative = the variation of C_n with the yaw rate - C_n^{r}.
   - Wing span = the reference wing span b.
2. *Inputs:*
   - WindAxesVel = the 3×1 vector of wind-axes velocities $[V_a \quad \alpha \quad \beta]$.
   - Control = the 4×1 vector of aerodynamic controls (actuators) in the following order: lift control (flap), pitch control (elevator), roll control (aileron), and yaw control (rudder) $[\delta_f \quad \delta_e \quad \delta_a \quad \delta_r]$.
   - AngularRates = the 3×1 vector of body angular rates $[p \quad q \quad r]$.
3. *Outputs:*
   - Cn = the current airplane yaw moment coefficient.
4. *Details:* The yaw moment coefficient is computed using the expression (25).

$$C_n = C_n^{\beta} \cdot \beta + C_n^{\delta_a} \cdot \delta_a + C_n^{\delta_r} \cdot \delta_r + \\ + \frac{b}{2V_a} \left(C_n^{p} \cdot p + C_n^{r} \cdot r \right) \tag{25}$$

5. *Usage:* The inputs should be provided by *Wind-axes Velocities* block in *Aerodynamics*, *Moments* block in *Equations of Motion*. Control input is provided by manual or automatic pilot. Block output is used by *Aerodynamic Force* block.

4.3 Atmosphere

The **Atmosphere** library folder includes blocks to estimate the air parameters and the wind effects.

4.3.1 Standard Atmosphere

The standard atmosphere block provides the air parameters at the current altitude. The block is presented in Fig. 45.

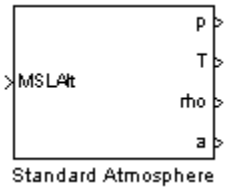


Figure 45: Standard Atmosphere Block

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - MSLAlt = the current altitude above Mean-Sea Level, in **[m]**.
3. *Outputs:*
   - p = static pressure, in **[Pa]**.
   - T = Outside-Air Temperature, in **[K]**.
   - rho = air density, in **[**kg/m^3**]**.
   - a = speed of sound, in **[m/s]**.
4. *Details:* The standard atmosphere block is using interpolation through look-up tables which provide air data for an altitude range of 0 to 86000 meters.
5. *Usage:* The block input can be provided by the *EGM-96* block in *Earth Model* section of the ***AeroSim*** library.

4.3.2 Background Wind

The block computes the background wind velocity components in body axes. It is represented in Fig. 46.

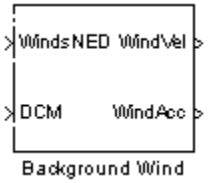


Figure 46: Background Wind Block

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - WindsNED = the $[3 \times 1]$ background wind velocity components in inertial frame (North-East-Down).
   - DCM = the $[3 \times 3]$ direction cosine matrix for inertial-to-body transformation.
3. *Outputs:*
   - WindVel = the $[3 \times 1]$ wind velocity components in body axes.
   - WindAcc = the $[3 \times 1]$ wind acceleration components in body axes.
4. *Details:* The block is applying a frame transformation from inertial (geographic) to body frame, using the rotation matrix provided. The numerical time derivative of the resulting velocity vector is then computed. This captures the effect of time-varying background wind which can be encountered in some weather conditions (wind shear, thermals, cyclones).
5. *Usage:* The input to this block can be provided as a constant, or from a weather model. The outputs of this block is used by the *Wind Force*, *Turbulence*, *Wind Shear* blocks in *Atmosphere* model, and by *Wind-axes Velocities* in *Aerodynamics*.

4.3.3 Turbulence

The block provides a *von Karman* turbulence model. It is shown in Fig. 47.

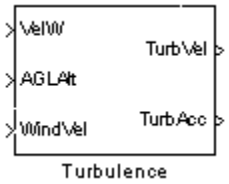


Figure 47: Turbulence Block

Block characteristics:

1. *Parameters:*
   - Sample time = the sample time for the white-noise sources.
2. *Inputs:*
   - VelW = the 3×1 vector of wind-axes velocities $[V_a \quad \alpha \quad \beta]$ (airspeed is in **[m/s]**).
   - AGLAlt = the current altitude Above Ground Level, in **[m]**.
   - WindVel = the 3×1 vector of background wind velocity in body axes, in **[m/s]**.
3. *Outputs:*
   - TurbVel = the 3×1 vector of turbulence velocities, in body axes, in **[m/s]**.
   - TurbAcc = the 3×1 vector of turbulence accelerations, in body axes, in **[m/s^2]**.
4. *Details:* The block is applying von Karman turbulence shaping filters for longitudinal, lateral, and vertical components to 3 white-noise sources. The filter parameters depend on background wind magnitude and current aircraft altitude.
5. *Usage:* The block gets its input from *Wind-axes Velocities* in *Aerodynamics*, *Ground Detection* in *Earth* model, and *Background Wind* in *Atmosphere* model. The outputs of the *Turbulence* block should go to *Wind Shear*, *Wind Force* blocks in *Atmosphere*.

4.3.4 Wind Shear

The block computes the angular rate effects caused by the variation in time/space of the background wind and turbulence velocities. It is represented in Fig. 48.

Figure 48: Wind Shear Block

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - WindAcc = the 3×1 vector of background wind accelerations.
   - TurbAcc = the 3×1 vector of turbulence accelerations.
   - Velocities = the 3×1 vector of aircraft velocities in body axes.
3. *Outputs:*
   - WindAngRates = the 3×1 vector of body angular rates caused by wind components.
   - WindAngAcc = the 3×1 vector of body angular accelerations caused by the wind components.
4. *Details:* The wind shear effects considered are the angular velocities and accelerations for pitch and yaw. The pitch rate due to wind is computed as shown in equation (26). The yaw rate due to wind is computed as shown in equation (27). The pitch and yaw accelerations are computed by taking numerical time derivatives of the angular rates.

$$q_{wind} = \frac{1}{u_{aircraft}} \cdot \frac{dw_{wind}}{dt} \tag{26}$$

$$r_{wind} = \frac{1}{u_{aircraft}} \cdot \frac{dv_{wind}}{dt} \tag{27}$$

5. *Usage:* The inputs to this block are provided by the *Background Wind* and *Turbulence* blocks in *Atmosphere* and *Forces* block in *Equations of Motion*. The outputs of this block are used by the *Wind Moment* block.

4.4 Complete Aircraft

The **Complete Aircraft** folder provides complete aircraft models built using **AeroSim** library blocks. These models are configurable via parameter files which should be in "mat" format. A template and sample Matlab scripts that can be used to generate this type of aircraft parameter files can be found in the **samples** directory, under the main **AeroSim** library directory.

4.4.1 6-DOF Aircraft Model - body-frame EOM

The block implements a nonlinear 6-degree-of-freedom aircraft dynamic model, using blocks provided in the **AeroSim** library. The equations of motion are implemented in body-axes. The model parameters are read from a user-configurable mat-file. The block is shown in Fig. 49.

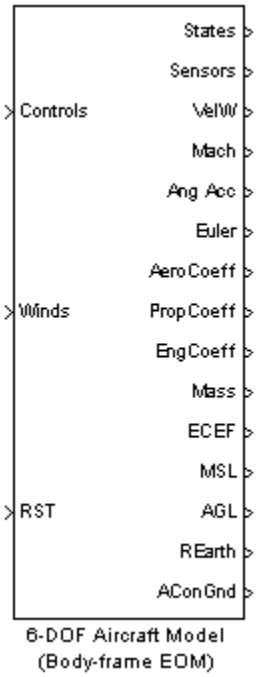


Figure 49: 6-DOF Aircraft Model Block

Block characteristics:

1. *Parameters:*

- Aircraft configuration file = the path and name of the aircraft parameter mat-file, provided as a string. For example, if the mat-file is **someairplane.mat**, and it is in the current directory, then we would use **'someairplane.mat'**.
- Initial velocities = the 3×1 vector of initial aircraft velocity components (groundspeed in body axes) $[u \quad v \quad w]^T$.
- Initial angular rates = the 3×1 vector of initial aircraft angular rates (in body axes)$[p \quad q \quad r]^T$.
- Initial attitude = the 4×1 vector of initial aircraft attitude provided as Euler-Rodrigues quaterions $[e_0 \quad e_x \quad e_y \quad e_z]^T$.
- Initial position = the 3×1 vector of initial aircraft location $[Lat \quad Lon \quad Alt]^T$, in [rad rad m].
- Initial fuel mass = the initial mass of the fuel quantity available on-board the aircraft, in kg.
- Initial engine speed = the initial engine shaft rotation speed, in rad/s.
- Ground altitude = the altitude of the terrain relative to mean-sea-level, at aircraft location, in meters.
- WMM coefficient file = the complete path to the magnetic model coefficient file, as a string - for example, $'c:\backslash udynamics\backslash projects\backslash AeroSim\backslash wmm.cof'$.
- Simulation date = the 3×1 vector of the calendar date in the format [Day Month Year].
- Sample time = the sample time at which the aircraft model will run.

2. *Inputs:*

   - Controls = the 7×1 vector of aircraft controls $[\ flap \quad elevator \quad aileron \quad rudder \quad throttle \quad mixture \quad ignition\ ]^T$ in [rad rad rad rad frac ratio bool].
   - Winds = the 3×1 vector of background wind velocities, in navigation frame $[W_N \quad W_E \quad W_D]^T$, in m/s.
   - RST = the integrator reset flag (can take values of 0 or 1, all integrators reset on rising-edge).

3. *Outputs:*

   - States = the 15×1 vector of aircraft states $[u \quad v \quad w \quad p \quad q \quad r \quad e_0 \quad ex \quad ey \quad ez \quad Lat \quad Lon \quad Alt \quad m_{fuel} \quad \Omega_{eng}]^T$.
   - Sensors = the 18×1 vector of sensor data $[Lat \quad Lon \quad Alt \quad V_N \quad V_E \quad V_D \quad a_x \quad a_y \quad a_z \quad p \quad q \quad r \quad p_{stat} \quad p_{dyn} \quad OAT \quad H_x \quad H_y \quad H_z]^T$.
   - VelW = the 3×1 vector of aircraft velocity in wind axes $[V_a \quad \beta \quad \alpha]^T$ in [m/s rad rad].
   - Mach = the current aircraft Mach number.
   - Angular Acc = the 3×1 vector of body angular accelerations $[\dot{p} \quad \dot{q} \quad \dot{r}]^T$.
   - Euler = the 3×1 vector of the attitude of the aircraft given in Euler angles $[\phi \quad \theta \quad \psi]^T$, in radians.
   - AeroCoeff = the 6×1 vector of aerodynamic coefficients $[C_D \quad C_Y \quad C_L \quad C_l \quad C_m \quad C_n]^T$, in rad^{-1}.
   - PropCoeff = the 3×1 vector of propeller coefficients $[J \quad C_T \quad C_P]^T$.
   - EngCoeff = the 5×1 vector of engine coefficients $[MAP \quad \dot{m}_{air} \quad \dot{m}_{fuel} \quad BSFC \quad P]^T$ given in [kPa kg/s kg/s g/(W*hr) W].
   - Mass = the current aircraft mass, in kg.
   - ECEF = the 3×1 vector of aircraft position in the Earth-centered, Earth-fixed frame $[X \quad Y \quad Z]^T$.
   - MSL = the aircraft altitude above mean-sea-level, in m.
   - AGL = the aircraft altitude above ground, in m.
   - REarth = the Earth equivalent radius, at current aircraft location, in m.
   - AConGnd = the aircraft-on-the-ground flag (0 if aircraft above ground, 1 if aircraft is on the ground).

4. *Details:* Most of the AeroSim blocks are used in this complete aircraft model. The blocks not included are the sensor models, actuator models, and pilot interface blocks (joystick input and graphical output).

   The model reads the aircraft parameters at the start of simulation from the specified mat-file. The mat-file is generated by the user for a particular aircraft model, by customizing and running an aircraft configuration script (Matlab program). An aircraft configuration script template as well as sample aircraft configuration scripts can be found in the **Samples** directory under the AeroSim tree.

5. *Usage:* The complete aircraft model can be used for modeling and simulating the dynamics of a conventional aircraft

with single piston engine and fixed pitch propeller. Specific aircraft parameters are set from a user-generated configuration script. Control inputs can be provided from a joystick for manual flight, or from an autopilot block for autonomous flight. Simulation output can be visualized using the Flight Simulator or FlightGear interface blocks.

4.4.2 6-DOF Aircraft Model - geodetic-frame EOM

The block implements a nonlinear 6-degree-of-freedom aircraft dynamic model, using blocks provided in the **AeroSim** library. The equations of motion are implemented in geodetic-frame. The model parameters are read from a user-configurable mat-file. The block is shown in Fig. 50.

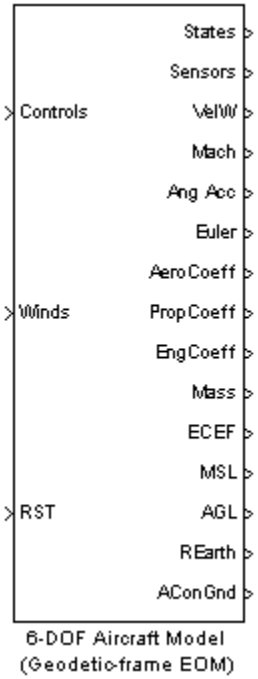


Figure 50: 6-DOF Aircraft Model Block

Block characteristics:

1. *Parameters:*

- Aircraft configuration file = the path and name of the aircraft parameter mat-file, provided as a string. For example, if the mat-file is **someairplane.mat**, and it is in the current directory, then we would use **'someairplane.mat'**.
- Initial position = the 3×1 vector of initial aircraft location $[Lat \quad Lon \quad Alt]^T$, in [rad rad m].
- Initial velocities = the 3×1 vector of initial aircraft velocity components in geodetic-frame $[V_N \quad V_E \quad V_D]^T$.
- Initial attitude = the 4×1 vector of initial aircraft attitude provided as Euler-Rodrigues quaterions $[e_0 \quad e_x \quad e_y \quad e_z]^T$.
- Initial angular rates = the 3×1 vector of initial aircraft angular rates (in body axes)$[p \quad q \quad r]^T$.
- Initial fuel mass = the initial mass of the fuel quantity available on-board the aircraft, in kg.
- Initial engine speed = the initial engine shaft rotation speed, in rad/s.
- Ground altitude = the altitude of the terrain relative to mean-sea-level, at aircraft location, in meters.
- WMM coefficient file = the complete path to the magnetic model coefficient file, as a string - for example, $'c:\backslash udynamics\backslash projects\backslash AeroSim\backslash wmm.cof'$.
- Simulation date = the 3×1 vector of the calendar date in the format [Day Month Year].
- Sample time = the sample time at which the aircraft model will run.

2. *Inputs:*

   - Controls = the 7×1 vector of aircraft controls $[\ flap \quad elevator \quad aileron \quad rudder \quad throttle \quad mixture \quad ignition]^T$ in [rad rad rad rad frac ratio bool].
   - Winds = the 3×1 vector of background wind velocities, in navigation frame $[W_N \quad W_E \quad W_D]^T$, in m/s.
   - RST = the integrator reset flag (can take values of 0 or 1, all integrators reset on rising-edge).

3. *Outputs:*

   - States = the 15×1 vector of aircraft states $[V_N \quad V_E \quad V_D \quad p \quad q \quad r \quad e_0 \quad ex \quad ey \quad ez \quad Lat \quad Lon \quad Alt \quad m_{fuel} \quad \Omega_{eng}]^T$.
   - Sensors = the 18×1 vector of sensor data $[Lat \quad Lon \quad Alt \quad V_N \quad V_E \quad V_D \quad a_x \quad a_y \quad a_z \quad p \quad q \quad r \quad p_{stat} \quad p_{dyn} \quad OAT \quad H_x \quad H_y \quad H_z]^T$.
   - VelW = the 3×1 vector of aircraft velocity in wind axes $[V_a \quad \beta \quad \alpha]^T$ in [m/s rad rad].
   - Mach = the current aircraft Mach number.
   - Angular Acc = the 3×1 vector of body angular accelerations $[\dot{p} \quad \dot{q} \quad \dot{r}]^T$.
   - Euler = the 3×1 vector of the attitude of the aircraft given in Euler angles $[\phi \quad \theta \quad \psi]^T$, in radians.
   - AeroCoeff = the 6×1 vector of aerodynamic coefficients $[C_D \quad C_Y \quad C_L \quad C_l \quad C_m \quad C_n]^T$, in rad^{-1}.
   - PropCoeff = the 3×1 vector of propeller coefficients $[J \quad C_T \quad C_P]^T$.
   - EngCoeff = the 5×1 vector of engine coefficients $[MAP \quad \dot{m}_{air} \quad \dot{m}_{fuel} \quad BSFC \quad P]^T$ given in [kPa kg/s kg/s g/(W*hr) W].
   - Mass = the current aircraft mass, in kg.
   - ECEF = the 3×1 vector of aircraft position in the Earth-centered, Earth-fixed frame $[X \quad Y \quad Z]^T$.
   - MSL = the aircraft altitude above mean-sea-level, in m.
   - AGL = the aircraft altitude above ground, in m.
   - REarth = the Earth equivalent radius, at current aircraft location, in m.
   - AConGnd = the aircraft-on-the-ground flag (0 if aircraft above ground, 1 if aircraft is on the ground).

4. *Details:* Most of the AeroSim blocks are used in this complete aircraft model. The blocks not included are the sensor models, actuator models, and pilot interface blocks (joystick input and graphical output).

   The model reads the aircraft parameters at the start of simulation from the specified mat-file. The mat-file is generated by the user for a particular aircraft model, by customizing and running an aircraft configuration script (Matlab program). An aircraft configuration script template as well as sample aircraft configuration scripts can be found in the **Samples** directory under the AeroSim tree.

5. *Usage:* The complete aircraft model can be used for modeling and simulating the dynamics of a conventional aircraft

with single piston engine and fixed pitch propeller. Specific aircraft parameters are set from a user-generated configuration script. Control inputs can be provided from a joystick for manual flight, or from an autopilot block for autonomous flight. Simulation output can be visualized using the Flight Simulator or FlightGear interface blocks.

4.4.3 6-DOF Aircraft Model - geodetic-frame EOM, no magnetic field

The block implements a nonlinear 6-degree-of-freedom aircraft dynamic model, using blocks provided in the **AeroSim** library. The equations of motion are implemented in geodetic-frame. The model parameters are read from a user-configurable mat-file. The block is shown in Fig. 51.

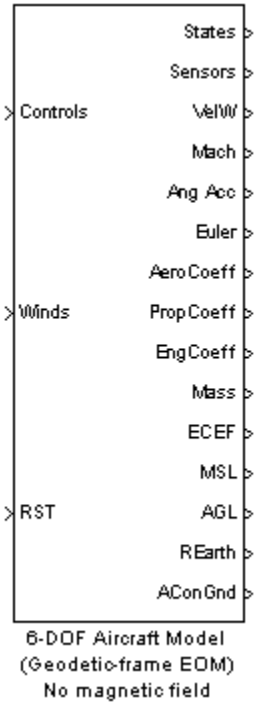


Figure 51: 6-DOF Aircraft Model Block

Block characteristics:

1. *Parameters:*

   - Aircraft configuration file = the path and name of the aircraft parameter mat-file, provided as a string. For example, if the mat-file is **someairplane.mat**, and it is in the current directory, then we would use **'someairplane.mat'**.
   - Initial position = the 3×1 vector of initial aircraft location $[Lat \quad Lon \quad Alt]^T$, in [rad rad m].
   - Initial velocities = the 3×1 vector of initial aircraft velocity components in geodetic-frame $[V_N \quad V_E \quad V_D]^T$.
   - Initial attitude = the 4×1 vector of initial aircraft attitude provided as Euler-Rodrigues quaterions $[e_0 \quad e_x \quad e_y \quad e_z]^T$.
   - Initial angular rates = the 3×1 vector of initial aircraft angular rates (in body axes)$[p \quad q \quad r]^T$.
   - Initial fuel mass = the initial mass of the fuel quantity available on-board the aircraft, in kg.
   - Initial engine speed = the initial engine shaft rotation speed, in rad/s.
   - Ground altitude = the altitude of the terrain relative to mean-sea-level, at aircraft location, in meters.
   - Sample time = the sample time at which the aircraft model will run.

2. *Inputs:*

- Controls = the 7×1 vector of aircraft controls $[\ flap \quad elevator \quad aileron \quad rudder \quad throttle \quad mixture \quad ignition\ ]^T$ in [rad rad rad rad frac ratio bool].
- Winds = the 3×1 vector of background wind velocities, in navigation frame $[W_N \quad W_E \quad W_D]^T$, in m/s.
- RST = the integrator reset flag (can take values of 0 or 1, all integrators reset on rising-edge).

3. *Outputs:*

   - States = the 15×1 vector of aircraft states $[V_N \quad V_E \quad V_D \quad p \quad q \quad r \quad e_0 \quad ex \quad ey \quad ez \quad Lat \quad Lon \quad Alt \quad m_{fuel} \quad \Omega_{eng}]^T$.
   - Sensors = the 15×1 vector of sensor data $[Lat \quad Lon \quad Alt \quad V_N \quad V_E \quad V_D \quad a_x \quad a_y \quad a_z \quad p \quad q \quad r \quad p_{stat} \quad p_{dyn} \quad OAT]^T$.
   - VelW = the 3×1 vector of aircraft velocity in wind axes $[V_a \quad \beta \quad \alpha]^T$ in [m/s rad rad].
   - Mach = the current aircraft Mach number.
   - Angular Acc = the 3×1 vector of body angular accelerations $[\dot{p} \quad \dot{q} \quad \dot{r}]^T$.
   - Euler = the 3×1 vector of the attitude of the aircraft given in Euler angles $[\phi \quad \theta \quad \psi]^T$, in radians.
   - AeroCoeff = the 6×1 vector of aerodynamic coefficients $[C_D \quad C_Y \quad C_L \quad C_l \quad C_m \quad C_n]^T$, in rad^{-1}.
   - PropCoeff = the 3×1 vector of propeller coefficients $[J \quad C_T \quad C_P]^T$.
   - EngCoeff = the 5×1 vector of engine coefficients $[MAP \quad \dot{m}_{air} \quad \dot{m}_{fuel} \quad BSFC \quad P]^T$ given in [kPa kg/s kg/s g/(W*hr) W].
   - Mass = the current aircraft mass, in kg.
   - ECEF = the 3×1 vector of aircraft position in the Earth-centered, Earth-fixed frame $[X \quad Y \quad Z]^T$.
   - MSL = the aircraft altitude above mean-sea-level, in m.
   - AGL = the aircraft altitude above ground, in m.
   - REarth = the Earth equivalent radius, at current aircraft location, in m.
   - AConGnd = the aircraft-on-the-ground flag (0 if aircraft above ground, 1 if aircraft is on the ground).

4. *Details:* Most of the AeroSim blocks are used in this complete aircraft model. The blocks not included are the sensor models, actuator models, and pilot interface blocks (joystick input and graphical output).

   The model reads the aircraft parameters at the start of simulation from the specified mat-file. The mat-file is generated by the user for a particular aircraft model, by customizing and running an aircraft configuration script (Matlab program). An aircraft configuration script template as well as sample aircraft configuration scripts can be found in the **Samples** directory under the AeroSim tree.

5. *Usage:* The complete aircraft model can be used for modeling and simulating the dynamics of a conventional aircraft with single piston engine and fixed pitch propeller. Specific aircraft parameters are set from a user-generated configuration script. Control inputs can be provided from a joystick for manual flight, or from an autopilot block for autonomous

flight. Simulation output can be visualized using the Flight Simulator or FlightGear interface blocks.

4.4.4 Simple Aircraft Model

The block implements a nonlinear 6-degree-of-freedom aircraft dynamic model, using blocks provided in the **AeroSim** library. The equations of motion are implemented in body-axes. The model parameters are read from a user-configurable mat-file. The model does not include wind effects. Earth is assumed spherical (constant radius, constant gravity). The block is shown in Fig. 52.

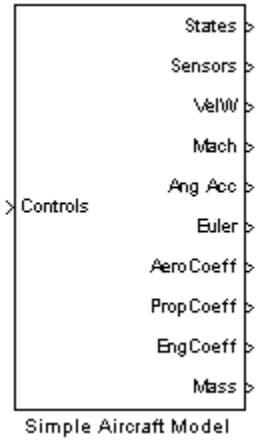


Figure 52: Simple Aircraft Model Block

Block characteristics:

1. *Parameters:*

   - Aircraft configuration file = the path and name of the aircraft parameter mat-file, provided as a string. For example, if the mat-file is **someairplane.mat**, and it is in the current directory, then we would use **'someairplane.mat'**.
   - Initial velocities = the 3×1 vector of initial aircraft velocity components (groundspeed in body axes) $[u \quad v \quad w]^T$.
   - Initial angular rates = the 3×1 vector of initial aircraft angular rates (in body axes)$[p \quad q \quad r]^T$.
   - Initial attitude = the 4×1 vector of initial aircraft attitude provided as Euler-Rodrigues quaterions $[e_0 \quad e_x \quad e_y \quad e_z]^T$.
   - Initial position = the 3×1 vector of initial aircraft location $[Lat \quad Lon \quad Alt]^T$, in [rad rad m].
   - Initial fuel mass = the initial mass of the fuel quantity available on-board the aircraft, in kg.
   - Initial engine speed = the initial engine shaft rotation speed, in rad/s.
   - Sample time = the sample time at which the aircraft model will run.

2. *Inputs:*

   - Controls = the 7×1 vector of aircraft controls $[flap \quad elevator \quad aileron \quad rudder \quad throttle \quad mixture \quad ignition]^T$ in [rad rad rad rad frac ratio bool].

3. *Outputs:*

   - States = the 15×1 vector of aircraft states $[u \quad v \quad w \quad p \quad q \quad r \quad e_0 \quad ex \quad ey \quad ez \quad Lat \quad Lon \quad Alt \quad m_{fuel} \quad \Omega_{eng}]^T$.

- Sensors = the 15×1 vector of sensor data $[Lat \quad Lon \quad Alt \quad V_N \quad V_E \quad V_D \quad a_x \quad a_y \quad a_z \quad p \quad q \quad r \quad p_{stat} \quad p_{dyn} \quad OAT]^T$.
- VelW = the 3×1 vector of aircraft velocity in wind axes $[V_a \quad \beta \quad \alpha]^T$ in [m/s rad rad].
- Mach = the current aircraft Mach number.
- Angular Acc = the 3×1 vector of body angular accelerations $[\dot{p} \quad \dot{q} \quad \dot{r}]^T$.
- Euler = the 3×1 vector of the attitude of the aircraft given in Euler angles $[\phi \quad \theta \quad \psi]^T$, in radians.
- AeroCoeff = the 6×1 vector of aerodynamic coefficients $[C_D \quad C_Y \quad C_L \quad C_l \quad C_m \quad C_n]^T$, in rad^{-1}.
- PropCoeff = the 3×1 vector of propeller coefficients $[J \quad C_T \quad C_P]^T$.
- EngCoeff = the 5×1 vector of engine coefficients $[MAP \quad \dot{m}_{air} \quad \dot{m}_{fuel} \quad BSFC \quad P]^T$ given in [kPa kg/s kg/s g/(W*hr) W].
- Mass = the current aircraft mass, in kg.

4. *Details:* The most important blocks of the AeroSim library are used in this simplified aircraft model. The blocks not included are the wind effects, the Earth models, the sensor models, actuator models, and pilot interface blocks (joystick input and graphical output).

   The model reads the aircraft parameters at the start of simulation from the specified mat-file. The mat-file is generated by the user for a particular aircraft model, by customizing and running an aircraft configuration script (Matlab program). An aircraft configuration script template as well as sample aircraft configuration scripts can be found in the **Samples** directory under the AeroSim tree.

5. *Usage:* The simple aircraft model can be used for modeling and simulating the dynamics of a conventional aircraft with single piston engine and fixed pitch propeller. Specific aircraft parameters are set from a user-generated configuration script. Control inputs can be provided from a joystick for manual flight, or from an autopilot block for autonomous flight. Simulation output can be visualized using the Flight Simulator or FlightGear interface blocks.

4.4.5 Glider Model

The block implements a basic un-powered nonlinear 6-degree-of-freedom aircraft model, using blocks provided in the **AeroSim** library. The equations of motion are implemented in body-axes. The model parameters are read from a user-configurable mat-file. The model does not include propulsion nor wind effects. Earth is assumed spherical (constant radius, constant gravity). The block is shown in Fig. 53.

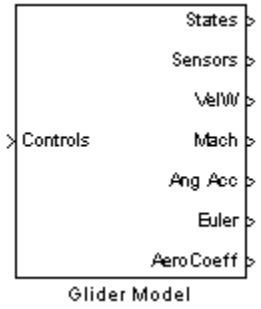


Figure 53: Glider Model Block

Block characteristics:

1. *Parameters:*

   - Aircraft configuration file = the path and name of the aircraft parameter mat-file, provided as a string. For example, if the mat-file is **someairplane.mat**, and it is in the current directory, then we would use **'someairplane.mat'**.
   - Initial velocities = the 3×1 vector of initial aircraft velocity components (groundspeed in body axes) $[u \quad v \quad w]^T$.
   - Initial angular rates = the 3×1 vector of initial aircraft angular rates (in body axes)$[p \quad q \quad r]^T$.
   - Initial attitude = the 4×1 vector of initial aircraft attitude provided as Euler-Rodrigues quaterions $[e_0 \quad e_x \quad e_y \quad e_z]^T$.
   - Initial position = the 3×1 vector of initial aircraft location $[Lat \quad Lon \quad Alt]^T$, in [rad rad m].
   - Sample time = the sample time at which the aircraft model will run.

2. *Inputs:*

   - Controls = the 4×1 vector of aerodynamic controls $[flap \quad elevator \quad aileron \quad rudder]^T$ in radians.

3. *Outputs:*

   - States = the 13×1 vector of aircraft states $[u \quad v \quad w \quad p \quad q \quad r \quad e_0 \quad ex \quad ey \quad ez \quad Lat \quad Lon \quad Alt]^T$.
   - Sensors = the 15×1 vector of sensor data $[Lat \quad Lon \quad Alt \quad V_N \quad V_E \quad V_D \quad a_x \quad a_y \quad a_z \quad p \quad q \quad r \quad p_{stat} \quad p_{dyn} \quad OAT]^T$.
   - VelW = the 3×1 vector of aircraft velocity in wind axes $[V_a \quad \beta \quad \alpha]^T$ in [m/s rad rad].
   - Mach = the current aircraft Mach number.
   - Angular Acc = the 3×1 vector of body angular accelerations $[\dot{p} \quad \dot{q} \quad \dot{r}]^T$.

- Euler = the 3×1 vector of the attitude of the aircraft given in Euler angles $[\phi \quad \theta \quad \psi]^T$, in radians.
- AeroCoeff = the 6×1 vector of aerodynamic coefficients $[C_D \quad C_Y \quad C_L \quad C_l \quad C_m \quad C_n]^T$, in rad^{-1}.

4. *Details:* The most basic blocks of the AeroSim library are used in this simplified aircraft model. The blocks not included are the propulsion model, the wind effects, the Earth models, the sensor models, actuator models, and pilot interface blocks (joystick input and graphical output).

   Since there is no propulsion nor fuel tank on-board, the aircraft inertia properties are assumed constant through-out the flight.

   The model reads the aircraft parameters at the start of simulation from the specified mat-file. The mat-file is generated by the user for a particular aircraft model, by customizing and running an aircraft configuration script (Matlab program). An aircraft configuration script template as well as sample aircraft configuration scripts can be found in the **Samples** directory under the AeroSim tree.

5. *Usage:* The simple aircraft model can be used for modeling and simulating the dynamics of a conventional aircraft with single piston engine and fixed pitch propeller. Specific aircraft parameters are set from a user-generated configuration script. Control inputs can be provided from a joystick for manual flight, or from an autopilot block for autonomous flight. Simulation output can be visualized using the Flight Simulator or FlightGear interface blocks.

4.4.6 Inertial Navigation System

The block implements the nonlinear 6-DOF INS (inertial navigation system) equations which integrate inertial measurements provided by an IMU (inertial measurement unit) to return the PVA (position velocity attitude) solution. The block is shown in Fig. 54.

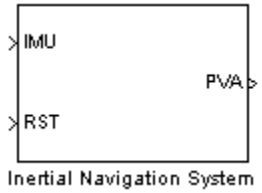


Figure 54: INS Block

Block characteristics:

1. *Parameters:*
   - Initial position = the 3×1 vector of initial aircraft location $[Lat \quad Lon \quad Alt]^T$, in [rad rad m].
   - Initial velocities = the 3×1 vector of initial aircraft velocity components in geodetic-frame $[V_N \quad V_E \quad V_D]^T$.
   - Initial attitude = the 4×1 vector of initial aircraft attitude provided as Euler-Rodrigues quaterions $[e_0 \quad e_x \quad e_y \quad e_z]^T$.
   - Sample time = the sample time at which the INS will run.
2. *Inputs:*
   - IMU = the 6×1 vector of inertial measurements (3 accelerations + 3 angular rates) in body axes $[a_x \quad a_y \quad a_z \quad p \quad q \quad r]^T$.
   - RST = the INS integrator reset flag (reset on rising-edge).
3. *Outputs:*
   - PVA = the INS states $[Lat \quad Lon \quad Alt \quad V_N \quad V_E \quad V_D \quad e_0 \quad e_x \quad e_y \quad e_z]^T$.
4. *Details:* The block implements the nonlinear 6-DOF equations of motion in geodetic-frame. It also includes a WGS-84 Earth geoid model.
5. *Usage:* The INS block can be used not in aircraft simulation, but it navigation systems, for GPS/INS integration, in conjunction with a navigation Kalman filter.

4.5 Earth

The **Earth** library folder includes blocks that model the Earth's shape, gravity, and magnetic field.

4.5.1 WGS-84

The block computes the local Earth radius and gravity at current aircraft location using the WGS-84 Earth model coefficients. The block is shown in Fig. 55.

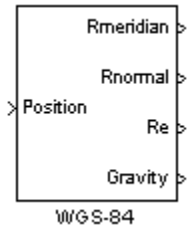


Figure 55: WGS-84 Block

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - Position = the 3×1 vector of geographic position [Lat Lon Alt], measured in [rad rad m].
3. *Outputs:*
   - Rmeridian = the meridian radius, in meters.
   - Rnormal = the normal radius, in meters.
   - Re = the equivalent radius, in meters.
   - Gravity = the local gravitational acceleration, in m/s^2.
4. *Details:* The WGS-84 coefficients and equations used by this block were obtained from [4], and they are provided below:

$$R_{meridian} = \frac{r_e(1-\varepsilon^2)}{(1-\varepsilon^2 \sin^2\phi)^{\frac{3}{2}}} \tag{28}$$

$$R_{normal} = \frac{r_e}{(1-\varepsilon^2 \sin^2\phi)^{\frac{1}{2}}} \tag{29}$$

$$R_{equiv} = \sqrt{R_{meridian} R_{normal}} \tag{30}$$

$$g = g_{WGS_0} \frac{1+g_{WGS_1} \sin^2\phi}{(1-\varepsilon^2 \sin^2\phi)^{\frac{1}{2}}} \tag{31}$$

   were the WGS-84 coefficients are:

   Equatorial radius $r_e = 6378137$ m

   First eccentricity $\varepsilon = 0.0818191908426$

   Gravity at equator $g_{WGS_0} = 9.7803267714\ m/s^2$

   Gravity formula constant $g_{WGS_1} = 0.00193185138639$

   and ϕ is the latitude at current aircraft location.

   Another WGS-84 reference can be found in [1].
5. *Usage:* The *WGS-84* block gets its input from the *Navigation* block in *Equations of Motion*. The block outputs are used in the same *Navigation* block and in the *Total Acceleration* block in the *Inertia* model.

4.5.2 EGM-96

The block computes the sea-level altitude with respect to the WGS-84 ellipsoid, using the EGM-96 geoid undulation model. The block is shown in Fig. 56.

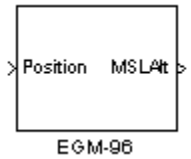


Figure 56: EGM-96 Block

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - Position = the 3×1 vector of geographic position [Lat Lon Alt], measured in [rad rad m].
3. *Outputs:*
   - MSLAlt = the altitude of the aircraft above Mean Sea Level, given in meters.
4. *Details:* The *EGM-96* block computes the altitude difference between the theoretical ellipsoid shape and the actual mean sea level (geoid undulation). This is caused by the non-uniformity of Earth's gravitational potential. The correction is performed using a 2-dimensional Latitude-Longitude look-up table with a resolution of 1 degree in both directions. The geoid undulation is then added to a -0.53 m WGS-84 correction and to the WGS-84 altitude computed by the aircraft equations of motion, to obtain the altitude of the aircraft above sea-level. More information about the EGM-96 model can be found at:

   http://cddisa.gsfc.nasa.gov/926/egm96/egm96.html.
5. *Usage:* The *EGM-96* block gets its input from the *Navigation* block in *Equations of Motion*. The block output is used in *Standard Atmosphere* block in the *Atmosphere* model, and in the *Ground Detection* block in the *Earth* model.

4.5.3 Ground Detection

The *Ground Detection* block computes the aircraft altitude Above Ground Level and sets a flag if it is zero. The block is shown in Fig. 57.

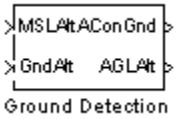


Figure 57: Ground Detection Block

Block characteristics:

1. *Parameters:* none.

2. *Inputs:*

   - MSLAlt = the altitude of the aircraft above Mean Sea Level.
   - GndAlt = the ground altitude with respect to MSL at current location.

3. *Outputs:*

   - AConGnd = the "Aircraft on the Ground" flag (0 or 1).
   - AGLAlt = the aircraft AGL altitude.

4. *Details:* The ground altitude should be supplied by the user as a constant or a look-up table of terrain elevation data. In both cases it should be measured with respect to the MSL and the unit of measure must match that of the MSL altitude.

   The "Aircraft on the Ground" flag can be used to stop the simulation when the aircraft has landed.

5. *Usage:* The *Ground Detection* block gets its inputs from the *EGM-96* block in the *Earth* model, and from the user. The outputs of this block are used by the *Turbulence* block in *Atmosphere* and by the *Navigation* block in the *Equations of Motion*.

4.5.4 WMM-2000

The *WMM-2000* block computes the Earth magnetic field components at current location using the Department of Defense World Magnetic Model 2000. The block is shown in Fig. 58.

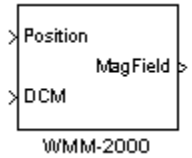


Figure 58: WMM-2000 Block

Block characteristics:

1. *Parameters:*
   - WMM coefficient file = the complete path to the magnetic model coefficient file, as a string - for example, $'c:\backslash udynamics\backslash projects\backslash AeroSim\backslash wmm.cof'$.
   - Simulation date = the 3×1 vector of the calendar date in the format [Day Month Year].
2. *Inputs:*
   - Position = the 3×1 vector of geographic position [Lat Lon Alt], measured in [rad rad m].
   - DCM = the 3×3 Direction Cosine Matrix for inertial-to-body transformation.
3. *Outputs:*
   - MagField = the 3×1 vector of magnetic field components in **body axes**, given in nT (nanoTesla).
4. *Details:* The block is using a hand-written S-function whose source code is provided (see **/aerosim/src/sfunwmm.c**). The code is based on the program **geomag.c** by *JOHN M. QUINN* and it is reading the interpolation coefficients from the text file **wmm.cof**. Due to the time variation of the Earth geomagnetic field, the data is reliable only for 5 years from the epoch date of the model, which is Jan. 1st, 2000. After that, an updated **wmm.cof** file will be required. Detailed information and the current **wmm.cof** file can be found on the NIMA website, at:

   http://164.214.2.59/GandG/ngdc-wmm2000.html.

   Please note that this model does not account for local spatial and temporal magnetic anomalies due to ground composition, solar activity, or electro-magnetic interference.
5. *Usage:* The *WMM-2000* block gets its inputs from the *Navigation* block in the *Equations of Motion* and from the *Body-Inertial DCM* block in *Transformations*. The block output can be provided to flight control systems as a simulated strap-down 3-axis magnetometer.

4.6 Equations of Motion

The **Equations of Motion** library folder includes blocks for all of the differential equations that describe the dynamics of the aircraft. These equations form the centerpiece of an aircraft dynamic model. There are two formulations for the equations of motion that are commonly used, and they are provided in two separate sub-folders within the AeroSim library. These are: the EOM with velocities in body axes (XYZ), and the EOM with aircraft velocities in geodetic frame (NED). From I/O point-of-view, both formulations provide the same inputs and outputs (for example, both will output velocities in body frame AND in geodetic frame) such that the designer can easily switch to a different EOM formulation after the aircraft model has already been built.

4.6.1 Total Acceleration

The block adds all accelerations applied to the airframe due to aerodynamics and propulsion, and returns the sum as body-axes components. The block is shown in Fig. 59.

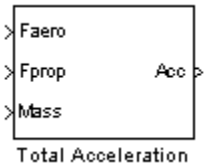


Figure 59: Total Acceleration Block

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - Faero = the 3×1 vector of aerodynamic forces in body axes $[F_{xaero} \quad F_{y_{aero}} \quad F_{z_{aero}}]^T$.
   - Fprop = the 3×1 vector of propulsion forces in body axes $\left[F_{x_{prop}} \quad F_{y_{prop}} \quad F_{z_{prop}}\right]^T$.
   - Mass = the current aircraft mass.
3. *Outputs:*
   - Acc = the 3×1 vector of total acceleration in body-axes $[a_x \quad a_y \quad a_z]^T$.
4. *Details:* The "total acceleration" includes everything except the gravity. Potential forces like gravity cannot be sensed by accelerometers since they affect all of the accelerometer sub-components in the same way. The total accelerations are computed simply as:

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}_{applied} = \frac{1}{M_{aircraft}} \cdot \left(\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix}_{aero} + \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix}_{prop} \right) \quad (32)$$

5. *Usage:* The block inputs are provided by the *Aerodynamic Force* block in *Aerodynamics*, the *Propulsion Force* block in *Propulsion* and the *Aircraft Inertia* block. The total acceleration output is used by the *Forces* block in the *Equations of Motion* and it can also be provided as aircraft model output in order to simulate a 3-axis accelerometer system.

4.6.2 Total Moment

The block computes the total moment applied to the airframe by the aerodynamics and propulsion. The moment is computed about the current aircraft CG location. The block is shown in Fig. 60.

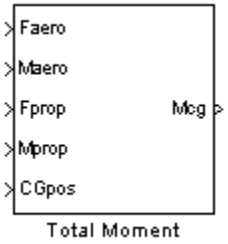


Figure 60: Total Moment Block

Block characteristics:

1. *Parameters:*
   - Aerodynamic force application point = the coordinates of the point at which the aerodynamic forces and moments are given, in body axes $r_{aero} = [X_{aero} \quad Y_{aero} \quad Z_{aero}]^T$.
   - Propulsion force application point = the coordinates of the point at which the propulsion forces and moments are given, in body axes $r_{prop} = [X_{prop} \quad Y_{prop} \quad Z_{prop}]^T$.
2. *Inputs:*
   - Faero = the 3×1 vector of aerodynamic forces in body axes $F_{aero} = [F_{x_{aero}} \quad F_{y_{aero}} \quad F_{z_{aero}}]^T$.
   - Maero = the 3×1 vector of aerodynamic moments in body axes $M_{aero} = [M_{x_{aero}} \quad M_{y_{aero}} \quad M_{z_{aero}}]^T$.
   - Fprop = the 3×1 vector of propulsion forces in body axes $F_{prop} = \left[F_{x_{prop}} \quad F_{y_{prop}} \quad F_{z_{prop}}\right]^T$.
   - Mprop = the 3×1 vector of propulsion moments in body axes $M_{prop} = \left[M_{x_{prop}} \quad M_{y_{prop}} \quad M_{z_{prop}}\right]^T$.
   - CGpos = the 3×1 vector of current CG position in body axes $r_{CG} = [X_{CG} \quad Y_{CG} \quad Z_{CG}]^T$.
3. *Outputs:*
   - Mcg = the 3×1 vector of total moment about the CG $M_{CG} = [M_x \quad M_y \quad M_z]^T$.
4. *Details:* The total moment is computed as shown below:

$$M_{CG} = (r_{aero} - r_{CG}) \times F_{aero} + M_{aero} + (r_{prop} - r_{CG}) \times F_{prop} + \\ + M_{prop} \tag{33}$$

5. *Usage:* The block inputs are provided by *Aerodynamic Force* and *Aerodynamic Moment* in *Aerodynamics*, by *Propulsion Force* and *Propulsion Moment* in *Propulsion* and by *Aircraft Inertia* block in the *Inertia* model. The block output is used by the *Moments* block in the *Equations of Motion*.

4.6.3 Body-frame EOM: Forces

The block implements the rigid-body 6 degree-of-freedom force equations that describe the time variation of the aircraft velocities. The block is shown in Fig. 61.

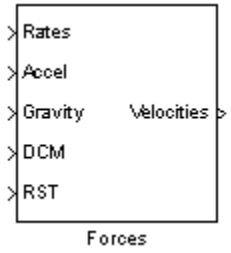


Figure 61: Forces Block

Block characteristics:

1. *Parameters:*
   - Initial velocities = the 3×1 vector of initial body velocities $[u_0 \quad v_0 \quad w_0]^T$.
2. *Inputs:*
   - Rates = the 3×1 vector of body angular rates $[p \quad q \quad r]^T$.
   - Accel = the 3×1 vector of body accelerations $[a_x \quad a_y \quad a_z]^T$.
   - Gravity = the gravitational acceleration as a scalar.
   - DCM = the 3×3 frame transformation matrix (from navigation (geodetic) to platform (body) frame.
   - RST = the integrator reset flag, which can be 0 or 1.
3. *Outputs:*
   - Velocities = the 3×1 vector of body velocities $[u \quad v \quad w]^T$.
4. *Details:* The force equations implemented can be found in [3] and they are presented below:

$$\begin{aligned} \dot{u} &= rv - qw + g_x + a_x \\ \dot{v} &= -ru + pw + g_y + a_y \\ \dot{w} &= qu - pv + g_z + a_z \end{aligned} \tag{34}$$

where:

$$\begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix} = R_{n2p} \cdot \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{35}$$

The initial condition for the integration of velocities is taken from the block parameters. If the reset flag changes from 0 to 1 during the simulation the integrator is reset back to the initial condition. The accelerations a_x, a_y, and a_z must include all the airframe loads (aerodynamics, propulsion, winds). Please note that the body velocities include the wind effects; in other words, they represent the ground speed of the airplane expressed in body axes.

5. *Usage:* The block takes its inputs from the *Moments* block in *Equations of Motion*, from *Total Acceleration* block in the *Inertia* model, and from the *WGS-84* Earth geoid model. The reset flag can be handled by the user. The block output is part of the aircraft state vector and it is used extensively throughout the aircraft model.

4.6.4 Body-frame EOM: Moments

The block integrates the rigid-body 6 degree-of-freedom moment equations to obtain the instantaneous body angular rates. The block is presented in Fig. 62.

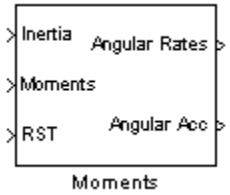


Figure 62: Moments Block

Block characteristics:

1. *Parameters:*
   - Initial angular rates = the 3×1 vector of initial body angular rates $[p_0 \quad q_0 \quad r_0]^T$.
2. *Inputs:*
   - Inertia = the 4×1 vector of current moments of inertia $[J_x \quad J_y \quad J_z \quad J_{xz}]^T$.
   - Moments = the 3×1 vector of airframe moments with respect to the CG $[L \quad M \quad N]^T$.
   - RST = the integrator reset flag, which can be 0 or 1.
3. *Outputs:*
   - Angular Rates = the 3×1 vector of body angular rates $[p \quad q \quad r]^T$.
   - Angular Acc = the 3×1 vector of body angular accelerations $[\dot{p} \quad \dot{q} \quad \dot{r}]^T$.
4. *Details:* The moment equations are implemented as shown in [3]. A summary of the equations is given below:

$$\begin{aligned} \dot{p} &= (c_1 r + c_2 p)\, q + c_3 L + c_4 N \\ \dot{q} &= c_5 pr - c_6 \left(p^2 - r^2\right) + c_7 M \\ \dot{r} &= (c_8 p - c_2 r)\, q + c_4 L + c_9 N \end{aligned} \tag{36}$$

   where the inertia coefficients c_1 - c_9 are computed using the *Inertia Coefficients* block available in the *Inertia* section of the **AeroSim** library. The moments should include all available loads (i.e. aerodynamics, propulsion, winds) and they should be given with respect to the current location of aircraft CG. The initial condition for the integration is taken from the block parameters. If the reset flag changes from 0 to 1 during the simulation the integrator is reset back to the initial condition.
5. *Usage:* The block gets its inputs from the *Aircraft Inertia* and *Total Moment* blocks in the *Inertia* model. The reset flag can be handled by the user. The block output is part of the aircraft state vector and it is used extensively throughout the aircraft model.

4.6.5 Body-frame EOM: Kinematics (Quaternions)

The *Kinematics (Quaternions)* block integrates the angular rates to obtain the aircraft attitude as quaternion representation. The block is presented in Fig. 63.

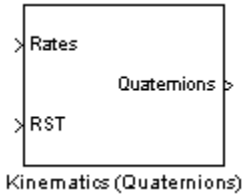


Figure 63: Kinematics (Quaternions) Block

Block characteristics:

1. *Parameters:*
   - Initial quaternions = the 4×1 vector of initial values for the quaternions $[e_{00} \quad e_{x0} \quad e_{y0} \quad e_{z0}]^T$.
2. *Inputs:*
   - Rates = the 3×1 vector of body angular rates $[p \quad q \quad r]^T$, given in rad/s.
   - RST = the integrator reset flag, which can be 0 or 1.
3. *Outputs:*
   - Quaternions = the 4×1 vector of quaternions $[e_0 \quad e_x \quad e_y \quad e_z]^T$.
4. *Details:* The kinematic equations are using the Euler-Rodrigues quaternions. This type of implementation is considered superior to the simple Euler angle equations, since quaternion equations are linear and the solution does not exhibit gimbal lock singularity. The equations, which are presented below, are described in more detail in [2].

$$\begin{bmatrix} \dot{e}_0 \\ \dot{e}_x \\ \dot{e}_y \\ \dot{e}_z \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix} \cdot \begin{bmatrix} e_0 \\ e_x \\ e_y \\ e_z \end{bmatrix} \tag{37}$$

   The initial condition for the integration is taken from the block parameters. If the reset flag changes from 0 to 1 during the simulation the integrator is reset back to the initial condition.
5. *Usage:* The block gets its input from the *Moments* block in the *Equations of Motion*. The reset flag can be handled by the user. The block output is part of the aircraft state vector and it serves as an input to the *Body-Inertial DCM From Quaternions* block in *Transformations*, whose output - the **Direction Cosine Matrix** is used extensively throughout the aircraft model.

4.6.6 Body-frame EOM: Kinematics (Euler Angles)

The *Kinematics (Euler Angles)* block integrates the angular rates to obtain the aircraft attitude as Euler angle representation. The block is presented in Fig. 64.

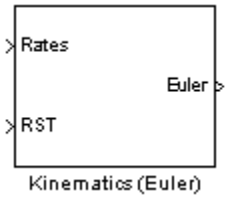


Figure 64: Kinematics (Euler Angles) Block

Block characteristics:

1. *Parameters:*
   - Initial Euler angles = the 3×1 vector of initial values for the Euler angles $[\phi_0 \quad \theta_0 \quad \psi_0]^T$.
2. *Inputs:*
   - Rates = the 3×1 vector of body angular rates $[p \quad q \quad r]^T$, given in rad/s.
   - RST = the integrator reset flag, which can be 0 or 1.
3. *Outputs:*
   - Euler = the 3×1 vector of Euler angles $[\phi \quad \theta \quad \psi]^T$.
4. *Details:* The kinematic equations are using the classic Euler angle representation.

$$\dot{\phi} = p + tan\theta(qsin\phi + rcos\phi) \tag{38}$$

$$\dot{\theta} = qcos\phi - rsin\phi \tag{39}$$

$$\dot{\psi} = \frac{qsin\phi + rcos\phi}{cos\theta} \tag{40}$$

   The initial condition for the integration is taken from the block parameters. If the reset flag changes from 0 to 1 during the simulation the integrator is reset back to the initial condition.
5. *Usage:* The block gets its input from the *Moments* block in the *Equations of Motion*. The reset flag can be handled by the user. The block output is part of the aircraft state vector and it serves as an input to the *Body-Inertial DCM From Euler Angles* block in *Transformations*, whose output - the **Direction Cosine Matrix** is used extensively throughout the aircraft model.

4.6.7 Body-frame EOM: Navigation

The block integrates the navigation equations to obtain the current aircraft position. It is represented in Fig. 65.

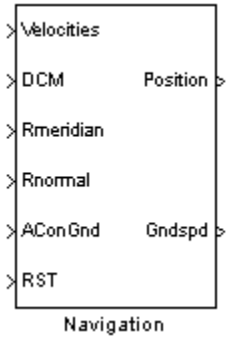


Figure 65: Navigation Block

Block characteristics:

1. *Parameters:*
   - Initial position = the 3×1 vector of initial geographic position $[Lat_0 \quad Lon_0 \quad Alt_0]^T$, where latitude and longitude are given in radians.
2. *Inputs:*
   - Velocities = the 3×1 vector of body-axes velocities $[u \quad v \quad w]^T$.
   - DCM = the 3×3 direction cosine matrix for inertial-to-body transformation.
   - Rmeridian = the meridian radius, in meters.
   - Rnormal = the normal radius, in meters.
   - AConGnd = the "Aircraft on the Ground" flag (0 or 1).
   - RST = the integrator reset flag, which can be 0 or 1.
3. *Outputs:*
   - Position = the 3×1 position vector $[Lat \quad Lon \quad Alt]^T$. Latitude and longitude are provided in radians.
   - Gndspd = the 3×1 ground speed vector components in geographic frame $[V_{North} \quad V_{East} \quad V_{Down}]^T$.
4. *Details:* The navigation equations implemented by the block are presented below. They are based on the navigation equations in [2].

$$\begin{bmatrix} V_{North} \\ V_{East} \\ V_{Down} \end{bmatrix} = DCM^T \cdot \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{41}$$

and geographic position is obtained from groundspeed as presented in [4]:

$$\dot{Lat} = \frac{V_{North}}{R_{meridian} + Alt} \tag{42}$$

$$\dot{Lon} = \frac{V_{East}}{(R_{normal} + Alt)\cos Lat} \tag{43}$$

$$\dot{Alt} = \begin{cases} -V_{Down}, AConGnd = 0 \\ 0, AConGnd = 1 \end{cases} \tag{44}$$

The altitude integration depends on the status of the "Aircraft on the Ground" flag.

5. *Usage:* The *Navigation* block gets its inputs from the *Forces* block in *Equations of Motion*, *Body-Inertial DCM* block in *Transformations*, *WGS-84* and *Ground Detection* blocks in the *Earth* model. The reset flag can be handled by the user. The position output is part of the aircraft state vector and it is used in the *WGS-84*, *EGM-96*, *WMM-2000* blocks in the *Earth* model, in the *ECEF Position* block in *Transformations*, and by the *FS Interface* block in *Pilot Interface*. The groundspeed vector components are not used in the model but they can be output as simulated GPS velocity measurements.

4.6.8 Geodetic-frame EOM: Position

The block integrates the aircraft inertial velocities to obtain the current aircraft position. It is represented in Fig. 66.

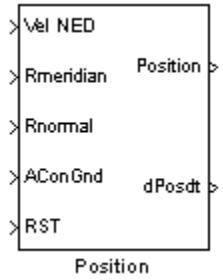


Figure 66: Position Block

Block characteristics:

1. *Parameters:*
   - Initial position = the 3×1 vector of initial geographic position $[Lat_0 \quad Lon_0 \quad Alt_0]^T$, where latitude and longitude are given in radians.
2. *Inputs:*
   - Vel NED = the 3×1 vector of inertial geodetic-frame velocities $[V_N \quad V_E \quad V_D]^T$.
   - Rmeridian = the meridian radius, in meters.
   - Rnormal = the normal radius, in meters.
   - AConGnd = the "Aircraft on the Ground" flag (0 or 1).
   - RST = the integrator reset flag, which can be 0 or 1.
3. *Outputs:*
   - Position = the 3×1 position vector $[Lat \quad Lon \quad Alt]^T$. Latitude and longitude are provided in radians.
   - dPosdt = the 3×1 vector of aircraft position derivatives $[\dot{Lat} \quad \dot{Lon} \quad \dot{Alt}]^T$.
4. *Details:* The position equations implemented by the block are presented below. They are based on the inertial navigation equations in [1].

$$\dot{Lat} = \frac{V_{North}}{R_{meridian} + Alt} \tag{45}$$

$$\dot{Lon} = \frac{V_{East}}{(R_{normal} + Alt)\cos Lat} \tag{46}$$

$$\dot{Alt} = \begin{cases} -V_{Down}, AConGnd = 0 \\ 0, AConGnd = 1 \end{cases} \tag{47}$$

   The altitude integration depends on the status of the "Aircraft on the Ground" flag.
5. *Usage:* The *Position* block gets its inputs from the *Velocity* block in *Equations of Motion*, *WGS-84* and *Ground Detection* blocks in the *Earth* model. The reset flag can be handled by the user. The position output is part of the aircraft state vector and it is used in the *WGS-84*, *EGM-96*, *WMM-2000* blocks in the *Earth* model, in the *ECEF Position* block in

Transformations, and by the *FS Interface* block in *Pilot Interface*. The position derivatives are used by the *Velocity* and the *Angular Rate* blocks in the *Equations of Motion*.

4.6.9 Geodetic-frame EOM: Velocity

The block integrates the accelerations applied to the airframe to obtain the inertial velocities of the aircraft in geodetic-frame. The block is shown in Fig. 67.

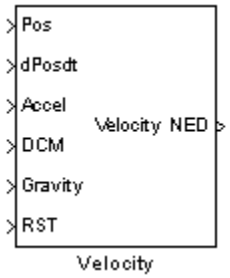


Figure 67: Velocity Block

Block characteristics:

1. *Parameters:*
   - Initial velocities = the 3×1 vector of initial geodetic-frame velocities $[V_{N0} \quad V_{E0} \quad V_{D0}]^T$.
2. *Inputs:*
   - Position = the 3×1 position vector $[Lat \quad Lon \quad Alt]^T$.
   - dPosdt = the 3×1 vector of aircraft position derivatives $[\dot{Lat} \quad \dot{Lon} \quad \dot{Alt}]^T$.
   - Accel = the 3×1 vector of body accelerations $[a_x \quad a_y \quad a_z]^T$.
   - DCM = the 3×3 frame transformation matrix (from navigation (geodetic) to platform (body) frame.
   - Gravity = the gravitational acceleration as a scalar.
   - RST = the integrator reset flag, which can be 0 or 1.
3. *Outputs:*
   - Velocity NED = the 3×1 vector of geodetic-frame velocities $[V_N \quad V_E \quad V_D]^T$.
4. *Details:* The velocity equations implemented can be found in [1] and they are presented below:

$$\begin{aligned} \dot{V_N} &= -(\dot{Lon} + 2\omega_{ie}) sin(Lat) V_E + \dot{Lat} V_D + f_N \\ \dot{V_E} &= (\dot{Lon} + 2\omega_{ie}) sin(Lat) V_N + (\dot{Lon} + 2\omega_{ie}) cos(Lat) V_D + f_E \\ \dot{V_D} &= -\dot{Lat} V_N - (\dot{Lon} + 2\omega_{ie}) cos(Lat) V_E + f_D + g \end{aligned} \tag{48}$$

   where:

$$\begin{bmatrix} f_N \\ f_E \\ f_D \end{bmatrix} = R_{p2n} \cdot \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \tag{49}$$

   The initial condition for the integration of velocities is taken from the block parameters. If the reset flag changes from 0 to 1 during the simulation the integrator is reset back to the initial condition. The accelerations a_x, a_y, and a_z must include all the airframe loads (aerodynamics, propulsion, winds).
5. *Usage:* The block takes its inputs from the *Position* block in *Equations of Motion*, from *Total Acceleration* block in the *Inertia* model, and from the *WGS-84* Earth geoid model. The reset flag can be handled by the user. The block output

is part of the aircraft state vector and it is used extensively throughout the aircraft model.

4.6.10 Geodetic-frame EOM: Attitude (Quaternions)

The *Attitude (Quaternions)* block is identical to the *Kinematics (Quaternions)* block and it integrates the angular rates to obtain the aircraft attitude. The block is presented in Fig. 68.

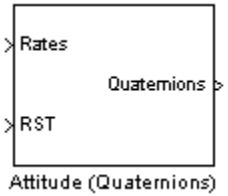


Figure 68: Attitude (Quaternions) Block

Block characteristics:

1. *Parameters:*
   - Initial quaternions = the 4×1 vector of initial values for the quaternions $[e_{00} \quad e_{x0} \quad e_{y0} \quad e_{z0}]^T$.
2. *Inputs:*
   - Rates = the 3×1 vector of body angular rates $[p \quad q \quad r]^T$, given in rad/s.
   - RST = the integrator reset flag, which can be 0 or 1.
3. *Outputs:*
   - Quaternions = the 4×1 vector of quaternions $[e_0 \quad e_x \quad e_y \quad e_z]^T$.
4. *Details:* The kinematic equations are using the Euler-Rodrigues quaternions. This type of implementation is considered superior to the simple Euler angle equations, since quaternion equations are linear and the solution does not exhibit gimbal lock singularity. The equations, which are presented below, are described in more detail in [2].

$$\begin{bmatrix} \dot{e}_0 \\ \dot{e}_x \\ \dot{e}_y \\ \dot{e}_z \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix} \cdot \begin{bmatrix} e_0 \\ e_x \\ e_y \\ e_z \end{bmatrix} \tag{50}$$

   The initial condition for the integration is taken from the block parameters. If the reset flag changes from 0 to 1 during the simulation the integrator is reset back to the initial condition.
5. *Usage:* The block gets its input from the *Angular Rate* block in the *Equations of Motion*. The reset flag can be handled by the user. The block output is part of the aircraft state vector and it serves as an input to the *Body-Inertial DCM From Quaternions* block in *Transformations*, whose output - the **Direction Cosine Matrix** is used extensively throughout the aircraft model.

4.6.11 Geodetic-frame EOM: Attitude (Euler Angles)

The *Attitude (Euler Angles)* block is identical to the *Kinematics (Euler Angles)* block and it integrates the angular rates to obtain the aircraft attitude. The block is presented in Fig. 69.

Figure 69: Attitude (Euler Angles) Block

Block characteristics:

1. *Parameters:*
   - Initial Euler angles = the 3×1 vector of initial values for the Euler angles $[\phi_0 \quad \theta_0 \quad \psi_0]^T$.
2. *Inputs:*
   - Rates = the 3×1 vector of body angular rates $[p \quad q \quad r]^T$, given in rad/s.
   - RST = the integrator reset flag, which can be 0 or 1.
3. *Outputs:*
   - Euler = the 3×1 vector of Euler angles $[\phi \quad \theta \quad \psi]^T$.
4. *Details:* The kinematic equations are using the classic Euler angle representation.

$$\dot{\phi} = p + tan\theta(qsin\phi + rcos\phi) \tag{51}$$

$$\dot{\theta} = qcos\phi - rsin\phi \tag{52}$$

$$\dot{\psi} = \frac{qsin\phi + rcos\phi}{cos\theta} \tag{53}$$

   The initial condition for the integration is taken from the block parameters. If the reset flag changes from 0 to 1 during the simulation the integrator is reset back to the initial condition.
5. *Usage:* The block gets its input from the *Moments* block in the *Equations of Motion*. The reset flag can be handled by the user. The block output is part of the aircraft state vector and it serves as an input to the *Body-Inertial DCM From Euler Angles* block in *Transformations*, whose output - the **Direction Cosine Matrix** is used extensively throughout the aircraft model.

4.6.12 Geodetic-frame EOM: Angular Rate

The block integrates the rigid-body 6 degree-of-freedom moment equations to obtain the instantaneous body angular rates. The block is presented in Fig. 70.

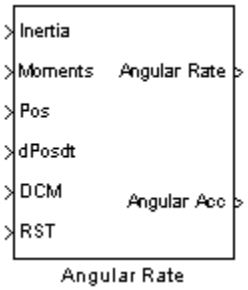


Figure 70: Angular Rate Block

Block characteristics:

1. *Parameters:*
   - Initial angular rates = the 3×1 vector of initial body angular rates $[p_0 \quad q_0 \quad r_0]^T$.
2. *Inputs:*
   - Inertia = the 4×1 vector of current moments of inertia $[J_x \quad J_y \quad J_z \quad J_{xz}]^T$.
   - Moments = the 3×1 vector of airframe moments with respect to the CG $[L \quad M \quad N]^T$.
   - Position = the 3×1 position vector $[Lat \quad Lon \quad Alt]^T$.
   - dPosdt = the 3×1 vector of aircraft position derivatives $[\dot{Lat} \quad \dot{Lon} \quad \dot{Alt}]^T$.
   - DCM = the 3×3 frame transformation matrix (from navigation (geodetic) to platform (body) frame.
   - RST = the integrator reset flag, which can be 0 or 1.
3. *Outputs:*
   - Angular Rates = the 3×1 vector of body angular rates $[p \quad q \quad r]^T$.
   - Angular Acc = the 3×1 vector of body angular accelerations $[\dot{p} \quad \dot{q} \quad \dot{r}]^T$.
4. *Details:* The moment equations are implemented as shown in [3]. A summary of the equations is given below:

$$\begin{aligned} \dot{p} &= (c_1 r + c_2 p)\, q + c_3 L + c_4 N - (\omega_{in}^p)_x \\ \dot{q} &= c_5 pr - c_6 \left(p^2 - r^2\right) + c_7 M - (\omega_{in}^p)_y \\ \dot{r} &= (c_8 p - c_2 r)\, q + c_4 L + c_9 N - (\omega_{in}^p)_z \end{aligned} \tag{54}$$

where the inertia coefficients c_1 - c_9 are computed using the *Inertia Coefficients* block available in the *Inertia* section of the **AeroSim** library. The moments should include all available loads (i.e. aerodynamics, propulsion, winds) and they should be given with respect to the current location of aircraft CG. The initial condition for the integration is taken from the block parameters. If the reset flag changes from 0 to 1 during the simulation the integrator is reset back to the

initial condition. The ω_{in}^{p} vector represents the Earth rotation rate in body-axes and it can be computed as follows:

$$\omega_{in}^{p} = R_{n2p} \begin{bmatrix} (\dot{Lon} + \omega_{ie})cos(Lat) \\ -\dot{Lat} \\ -(\dot{Lon} + \omega_{ie})sin(Lat) \end{bmatrix} \tag{55}$$

5. *Usage:* The block gets its inputs from the *Aircraft Inertia* and *Total Moment* blocks in the *Inertia* model, and from the *Position* block in *Equations of Motion*. The reset flag can be handled by the user. The block output is part of the aircraft state vector and it is used extensively throughout the aircraft model.

4.7 Inertia

The **Inertia** library folder includes blocks that model the evolution of the aircraft inertia parameters, including mass, CG position, and moments of inertia. Also, the same folder includes the blocks that sum-up all the forces and moments provided by the aerodynamic, propulsion, and atmospheric models to obtain a single force and a single moment that can be used in the equations of motion.

4.7.1 Aircraft Inertia

The block integrates the instantaneous fuel consumption to obtain the current aircraft inertia parameters: mass, CG location, and moments of inertia. The block is shown in Fig. 71.

Figure 71: Aircraft Inertia Block

Block characteristics:

1. *Parameters:*
   - Initial fuel mass = the initial value for the fuel quantity available on board the aircraft.
   - Empty mass = the aircraft mass without fuel.
   - Gross mass = the aircraft mass with full fuel tank.
   - Empty CG location = the 1×3 vector of the CG location for the aircraft without fuel $[X \quad Y \quad Z]$, in body axes.
   - Gross CG location = the 1×3 vector of the CG location for the aircraft with full fuel tank $[X \quad Y \quad Z]$, in body axes.
   - Empty moments of inertia = the 1×4 vector of the moments of inertia for the aircraft without fuel $[J_x \quad J_y \quad J_z \quad J_{xz}]$, with respect to the empty CG location.
   - Gross moments of inertia = the 1×4 vector of the moments of inertia for the aircraft with full fuel tank $[J_x \quad J_y \quad J_z \quad J_{xz}]$, with respect to the gross CG location.
2. *Inputs:*
   - FuelCon = the instantaneous fuel consumption rate.
   - RST = the integrator reset flag, which can be 0 or 1.
3. *Outputs:*
   - Mass = the current aircraft mass.
   - CGpos = the 3×1 CG position vector $[X \quad Y \quad Z]$, in body axes.
   - Inertia = the 4×1 moments of inertia vector $[J_x \quad J_y \quad J_z \quad J_{xz}]$ with respect to the current CG location.
   - OutofFuel = the "Out of Fuel" flag, which can be 0 or 1.
4. *Details:* The initial condition for the integration is taken from the block parameters. If the reset flag changes from 0 to 1 during the simulation the integrator is reset back to the initial condition. The inertia parameters are computed using linear interpolation between the empty and the gross aircraft inertia parameters, based on the current aircraft mass. When the fuel quantity obtained from integration of fuel consumption rate reaches zero the fuel integrator saturates to prevent it from dropping to negative values, and the "Out of Fuel" flag gets set to 1. The flag can then be used to command engine shut-down.

5. *Usage:* The block input should be provided by the engine model. The reset flag can be handled by the user. The mass output is used by the *Total acceleration* block in the *Inertia* model and by the *Wind Force* block in the *Atmosphere* model. The CG position output is used by the *Total moment* block in the *Inertia* model. The *Inertia* output is used by the *Moments* equations in the *Equations of Motion* and by the *Wind Moment* block in the *Atmosphere* model.

4.7.2 Inertia Coefficients

The block computes the inertia coefficients c_1 - c_9 that are used in the *Moments* equations. The block is presented in Fig. 72.

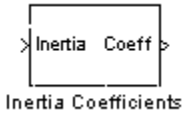


Figure 72: Inertia Coefficients Block

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - Inertia = the 4×1 moments of inertia vector $[J_x \quad J_y \quad J_z \quad J_{xz}]$ with respect to the current CG location.
3. *Outputs:*
   - Coeff = the 9×1 vector of inertia coefficients.
4. *Details:* The inertia coefficients are presented in [3] as a convenient way of bypassing the matrix inversion that would normally be required for solving the Moment Equations. The method is valid for most aircraft due to the symmetry about the mid plane, but it will not work for asymmetric configurations such as Burt Rutan's **Boomerang** since for these type of airplanes the coupling moments of inertia J_{xy} and J_{yz} are not zero. The inertia coefficients expressions are presented below:

$$
\begin{aligned}
c_1 &= \frac{(J_y - J_z)J_z - J_{xz}^2}{\Gamma} \\
c_2 &= \frac{(J_x - J_y + J_z)J_{xz}}{\Gamma} \\
c_3 &= \frac{J_z}{\Gamma} \\
c_4 &= \frac{J_{xz}}{\Gamma} \\
c_5 &= \frac{J_z - J_x}{J_y} \\
c_6 &= \frac{J_{xz}}{J_y} \\
c_7 &= \frac{1}{J_y} \\
c_8 &= \frac{J_x(J_x - J_y) + J_{xz}^2}{\Gamma} \\
c_9 &= \frac{J_x}{\Gamma}
\end{aligned}
\tag{56}
$$

where

$$\Gamma = J_x J_z - J_{xz}^2 \tag{57}$$

5. *Usage:* The block is referenced in two places - in the *Moments* block in the *Equations of Motion* and in the *Wind Moment* block in the *Atmosphere* model.

4.8 Math

This library folder includes all of the math functions that are referenced by blocks in the **AeroSim** library.

4.8.1 Cross Product

The block computes the cross product of two 3×1 vectors, and it is presented in Fig. 73.

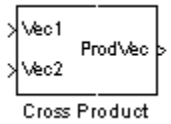


Figure 73: Cross Product Block

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - Vec1 = the first 3×1 vector.
   - Vec2 = the second 3×1 vector.
3. *Outputs:*
   - ProdVec = the 3×1 cross product result.
4. *Details:* The block computes the product:

$$\vec{P} = \vec{V}_1 \times \vec{V}_2 \tag{58}$$

5. *Usage:* The block is referenced in *Total Moment* block in the *Inertia* model.

4.8.2 Normalization

The block normalizes a vector of arbitrary dimension by dividing each vector element by the vector norm (magnitude). It is shown in Fig. 74.

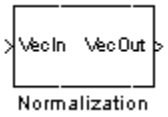


Figure 74: Normalization Block

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - VecIn = the input column vector of dimension $n \times 1$.
3. *Outputs:*
   - VecOut = the output normal vector of the same dimension $n \times 1$.
4. *Details:* The block implementation is simply:

$$\vec{V}_{out} = \frac{\vec{V}_{in}}{\|\vec{V}_{in}\|} \tag{59}$$

5. *Usage:* The block is referenced in *Kinematics* in the *Equations of Motion* in order to normalize the quaternion vector after it has been computed.

4.8.3 Vector Norm

The block computes the 2-norm or magnitude of a vector of arbitrary dimension. It is presented in Fig. 75.

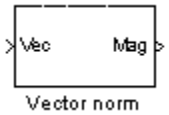


Figure 75: Vector Norm Block

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - Vec = the input column vector, of dimension $n \times 1$.
3. *Outputs:*
   - Mag = the magnitude (scalar).
4. *Details:* The magnitude is computed simply using the relationship:
$$\|\vec{V}\| = \sqrt{\vec{V}^T \vec{V}} \tag{60}$$
5. *Usage:* The block is referenced by the *Normalization* block in *Math*, and by the *Wind-axes Velocities* in *Aerodynamics*.

4.8.4 Non-zero Sign

The block implements the sign function with strictly-positive output for zero input.

Figure 76: Non-zero Sign Block

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - In = a real number.
3. *Outputs:*
   - Out = the non-zero sign of the input number (-1 or 1).
4. *Details:* The block implements the following branched function:
$$f(x) = \begin{cases} 1, & for \quad x \geq 0 \\ -1, & for \quad x < 0 \end{cases} \tag{61}$$
5. *Usage:* The block is referenced by the *Zero Offset* block in *Math*.

4.8.5 Zero Offset

The block prevents division-by-zero errors by outputting a small non-zero value equal to the specified tolerance if the block input value falls below this tolerance threshold. If the input is outside the sphere specified by this tolerance, then the output will match the input exactly. The block is presented in Fig. 77.

Figure 77: Zero Offset Block

Block characteristics:

1. *Parameters:*
   - Tolerance = the tolerance about origin for which the block will hold output.
2. *Inputs:*
   - In = a real number.
3. *Outputs:*
   - Out = a non-zero real number.
4. *Details:* The block is implementing the following function:

$$f(x) = \begin{cases} x, & for \quad \|x\| \geq tol \\ tol, & for \quad \|x\| < tol \end{cases} \tag{62}$$

5. *Usage:* The block is referenced from *Wind Shear* block in *Atmosphere* to prevent division-by-zero.

4.8.6 pi Bound

The *pi Bound* block clamps an angle input to the interval $[-\pi, \pi]$.
The block is shown in Fig. 78.

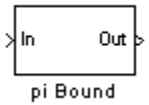


Figure 78: pi Bound Block

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - Input = an angle in **radians**.
3. *Outputs:*
   - Output = an angle in **radians**, inside the interval $[-\pi, \pi]$.
4. *Details:* When the input angle crosses a multiple of π, the output angle wraps back to $-\pi$. Similarly, then the input angle crosses the lower bound, the output wraps back to π. This is accomplished using the **mod** function, as shown below:
$$Output = mod(Input + \pi, 2\pi) - \pi \qquad (63)$$
5. *Usage:* The block is referenced in *Euler Angles* block in *Transformations* to limit the roll and the pitch angle to the $[-\pi, \pi]$ interval.

4.8.7 2pi Bound

The *2pi Bound* block clamps an angle input to the interval $[0, 2\pi]$. The block is shown in Fig. 79.

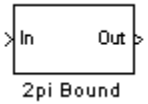


Figure 79: 2pi Bound Block

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - Input = an angle in **radians**.
3. *Outputs:*
   - Output = an angle in **radians**, inside the interval $[0, 2\pi]$.
4. *Details:* When the input angle crosses a multiple of 2π, the output angle wraps back to 0. Similarly, then the input angle crosses the lower bound, the output wraps back to 2π. This is accomplished using the **mod** function, as shown below:

$$Output = mod(Input, 2\pi) \tag{64}$$

5. *Usage:* The block is referenced in *Euler Angles* block in *Transformations* to limit the heading angle to the $[0, 2\pi]$ interval.

4.9 Pilot Interface

The **Pilot Interface** library folder includes input and output drivers for devices outside the **Matlab/Simulink** environment, through which the user can interact with the aircraft model.

4.9.1 FS Interface

The block provides an interface to **Microsoft Flight Simulator** for a real-time visual display of the aircraft. The block is shown in Fig. 80.

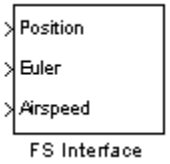


Figure 80: FS Interface Block

Block characteristics:

1. *Parameters:*
   - Sample time = the sample time at which position and attitude data will be sent to **Microsoft Flight Simulator**.
2. *Inputs:*
   - Position = the 3×1 vector of geographic position [Lat Lon Alt] in **[rad rad m]**.
   - Euler = the 3×1 vector of Euler angles $[\phi \quad \theta \quad \psi]^T$ in **radians**.
   - Airspeed = the current aircraft airspeed, in **m/s**.
3. *Outputs:* no output signals to Simulink, only visual output in **Microsoft Flight Simulator**.
4. *Details:* The block is implemented using the CMEX S-function **sfunflightsim.dll**. The mechanism through which **MSFS** state variables are accessed and written is by using Inter-Process Communication provided by **FSUIPC** utility by Peter L. Dowson. The module **fsuipc.dll** has to be installed in **Microsoft Flight Simulator**. The **MSFS** state variables are then accessed using the functions available in the library **IPCuser.c**. Memory addresses of these state variables are defined in **FSUIPC_User.h**.

   The S-function provided with the *FS Interface* block is overwriting the aircraft position, attitude, and airspeed variables in **Microsoft Flight Simulator** to provide a visual display of the aircraft behavior. To prevent interference with **MSFS**' own aircraft dynamics, **MSFS** should be run in SLEW mode. When starting the simulation, the *FS Interface* block will enable the SLEW mode automatically. The disadvantage is that some instruments in the aircraft cockpit panel displayed in **MSFS** are not updated in this mode. However, the main objective - displaying an outside view of the aircraft - is accomplished, see Fig. 81.

   Both **Matlab/Simulink** and **Microsoft Flight Simulator** are computationally-intensive applications. Although it is possible to run both on the same computer, we recommend executing them on separate machines. The way this can be done is by making use of the **WideFS** utility by Peter L. Dowson. This Client/Server application allows remote access to **MSFS** state variables using **FSUIPC**. This requires the installation of **wideserver.dll** in addition to **fsuipc.dll** on the **MSFS** machine, and the execution of **wideclient.exe** on

Figure 81: Aerosonde in a simulated weather recon flight

the **Matlab/Simulink** machine.

Latest versions of **FSUIPC** and **WIDEFS** can be downloaded at:

http://www.flightsim.com.

5. *Usage:* The *FS Interface* block gets its inputs from the *Navigation* block in *Equations of Motion*, from *Euler Angles* block in *Transformations* and from *Wind-axes Velocities* block in *Aerodynamics*. The only block output is visual and it requires the presence of **Microsoft Flight Simulator** on the local machine or on a secondary machine on the local network.

4.9.2 FlightGear Interface

The block provides an interface to **FlightGear Flight Simulator** 0.9.2 for a real-time visual display of the aircraft. The block is shown in Fig. 82.

Figure 82: FlightGear Interface Block

Block characteristics:

1. *Parameters:*
   - Host name = the name or IP address of the machine on which FlightGear is running, provided as a string. For example, if we want to run both Matlab and FlightGear on the same computer (not recommended performance-wise) we would use **'localhost'**.Otherwise, we can type a **'hostname'** or an IP address **'192.168.0.247'**.
   - Port = the number of the port on the host machine to which this interface will attempt to connect to. The default setting is port 5500.
   - Sample time = the sample time at which position and attitude data will be sent to **FlightGear Flight Simulator**.
2. *Inputs:*
   - Position = the 3×1 vector of geographic position [Lat Lon Alt] in **[rad rad m]**.
   - Euler = the 3×1 vector of Euler angles $[\phi \quad \theta \quad \psi]^T$ in **radians**.
   - Airspeed = the current aircraft airspeed, in **m/s**.
3. *Outputs:* no output signals to Simulink, only visual output in **FlightGear Flight Simulator**.
4. *Details:* The block is implemented using the CMEX S-function **sfunflightgear092.dll**. The S-function is fully reentrant - that is, multiple instances of this block can be executed in Simulink at the same time.
   Data is being sent to **FlightGear** using the UDP network protocol, on port 5500. To set-up FlightGear for accepting external aircraft model over the UDP connection, you must run its executable with the following command line arguments:

```
%FG_ROOT%\BIN\FGFS.EXE --disable-splash-screen
    --native-fdm=socket,in,30,,5500,udp --fdm=external
```

A screen capture of the visual output is shown in Fig. 83.

Both **Matlab/Simulink** and **FlightGear Flight Simulator** are computationally-intensive applications. Although it is possible to run both on the same computer, we recommend executing them on separate machines. The version of **FlightGear Flight Simulator** supported by this S-function is 0.9.2. Other **FlightGear Flight Simulator** releases can be used,

Figure 83: FlightGear output from the AeroSim FlightGear Interface block

but in this case the S-function **sfunflightgear092.dll** will have to be recompiled using the **net_fdm.hxx** from the **FlightGear Flight Simulator** source tree. Latest version can be downloaded at: **http://www.flightgear.org**.

5. *Usage:* The *FlightGear Interface* block gets its inputs from the *Navigation* block in *Equations of Motion*, from *Euler Angles* block in *Transformations* and from *Wind-axes Velocities* block in *Aerodynamics*. The only block output is visual and it requires the presence of **FlightGear Flight Simulator** on a machine on the local network.

4.9.3 FlightGear 0.9.4 Interface

The block provides an interface to **FlightGear Flight Simulator** 0.9.4 for a real-time visual display of the aircraft. The block is shown in Fig. 84.

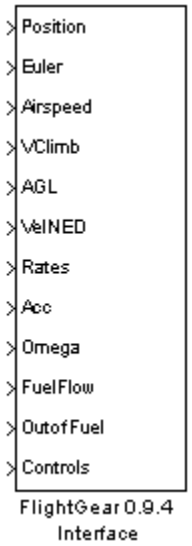


Figure 84: FlightGear 0.9.4 Interface Block

Block characteristics:

1. *Parameters:*

   - Host name = the name or IP address of the machine on which FlightGear is running, provided as a string. For example, if we want to run both Matlab and FlightGear on the same computer (not recommended performance-wise) we would use **'localhost'**.Otherwise, we can type a **'hostname'** or an IP address **'192.168.0.247'**.
   - Port = the number of the port on the host machine to which this interface will attempt to connect to. The default setting is port 5500.
   - Sample time = the sample time at which position and attitude data will be sent to **FlightGear Flight Simulator**.

2. *Inputs:*

   - Position = the 3×1 vector of geographic position [Lat Lon Alt] in **[rad rad m]**.
   - Euler = the 3×1 vector of Euler angles $[\phi \quad \theta \quad \psi]^T$ in **radians**.
   - Airspeed = the current aircraft airspeed, in **m/s**.
   - VClimb = the current climb rate, in **m/s**.
   - AGL = the altitude of the aircraft above ground, in **m**.
   - VelNED = the 3×1 vector of North East Down ground-speed components, in **m/s**.
   - Rates = the 3×1 vector of body angular rates, in **rad/s**.
   - Acc = the 3×1 vector of body linear accelerations, in **m/**s^2.
   - Omega = the propeller rotation speed, in **rad/s**.
   - FuelFlow = the instantaneous mass fuel flow, in **grams/hr**.

- OutofFuel = the out-of-fuel flag can be set to 1 to disable the engine sound in FlightGear. Otherwise, set it to 0.
- Controls = the 7×1 vector of aircraft controls, of which only the first four (the aerodynamic controls) are sent. The order in which they should be provided is [flap, elevator, aileron, rudder]. The control inputs should be normalized, such that their full range is -1 to 1.

3. *Outputs:* no output signals to Simulink, only visual output in **FlightGear Flight Simulator**.

4. *Details:* The block is implemented using the CMEX S-function **sfunflightgear094.dll**. For more details, consult the description of the original *FlightGear Interface* block in subsection 4.9.2.

5. *Usage:* The *FlightGear Interface* block gets its inputs from the *Navigation* block in *Equations of Motion*, from *Euler Angles* block in *Transformations* and from *Wind-axes Velocities* block in *Aerodynamics*. The only block output is visual and it requires the presence of **FlightGear Flight Simulator** on a machine on the local network.

4.9.4 Joystick Interface

The *Joystick Interface* block provides access to joystick position and buttons using the standard **Microsoft Windows** joystick class. The block is presented in Fig. 85.

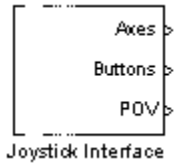


Figure 85: Joystick Interface Block

Block characteristics:

1. *Parameters:*
   - Joystick ID = the joystick identifier is normally set to 1, but for reading a second joystick, it can be set to 2.
   - Sample time = the time interval at which joystick information is sampled.
2. *Inputs:* no input signals from Simulink, the inputs are provided by the operating system.
3. *Outputs:*
   - Axes = the 6×1 vector of joystick position for all 6 axes. On each axis the position is provided as an unsigned integer from 0 to 65535.
   - Buttons = the 32×1 vector of button flags. For each button the flag is 0 if the button is not pressed, and 1 if the button is pressed.
   - POV = the position of the point-of-view hat switch. The output is an unsigned integer from 0 to 65535.
4. *Details:* The block is using the CMEX S-function **sfunjoy.dll**. The S-function is fully reentrant - that is, multiple instances of this block can be executed in Simulink at the same time. Thus, with two blocks of this type, the Simulink model can have access to two separate joysticks simultaneously.
5. *Usage:* The joystick block can be used to provide user inputs for control surfaces deflections, throttle setting and other aircraft commands.

4.9.5 CH F-16 Combat Stick

The *CH F-16 Combat Stick* is a custom joystick interface for the **F-16 Combat Stick** by **CH Products** (Fig. 86), based on the generic *Joystick Interface* block provided in the **AeroSim** library. The block is shown in Fig. 87.

Figure 86: CH F-16 Combat Stick

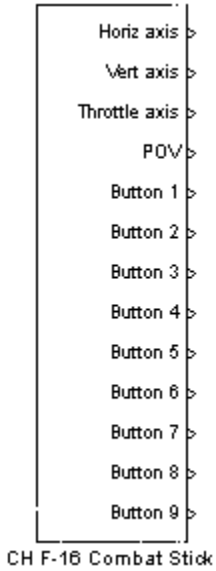


Figure 87: CH F-16 Combat Stick Block

Block characteristics:

1. *Parameters:*

   - Joystick ID = the joystick identifier is normally set to 1, but for reading a second joystick, it can be set to 2.

- Sample time = the time interval at which joystick information is sampled.

2. *Inputs:* no input signals from Simulink, the inputs are provided by the operating system.

3. *Outputs:*

   - Horiz axis = the joystick position on the horizontal axis (0 to 65535).
   - Vert axis = the joystick position on the vertical axis (0 to 65535).
   - Throttle axis = the throttle wheel position (0 to 65535).
   - POV = the POV hat position (0=center, 1=up, -1=down, 2=right, -2=left).
   - Button 1 through 9 = the joystick button states (0=released, 1=pressed).

4. *Details:* The block is using the same CMEX S-function as the generic *Joystick Interface* block, **sfunjoy.dll**, but it is customizing the outputs and providing only those axes and buttons that are physically available with the **F-16 Combat Stick** device.

5. *Usage:* The block can be used to provide user inputs for control surfaces deflections, throttle setting and other aircraft commands.

4.9.6 R/C Transmitter Interface

The *R/C Transmitter Interface* is a custom joystick interface for use with R/C radio transmitters, based on the generic *Joystick Interface* block provided in the **AeroSim** library.

The block is shown in Fig. 88.

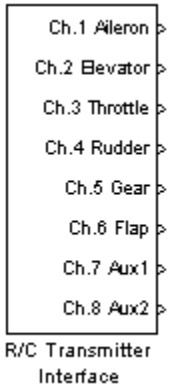


Figure 88: R/C Transmitter Interface Block

Block characteristics:

1. *Parameters:*
   - Joystick ID = the joystick identifier is normally set to 1, but for reading a second joystick, it can be set to 2.
   - Sample time = the time interval at which joystick information is sampled.
2. *Inputs:* no input signals from Simulink, the inputs are provided by the operating system.
3. *Outputs:*
   - Ch.1 Aileron = The channel 1 output (aileron or roll cyclic on Futaba radios), normalized to [-1, 1].
   - Ch.2 Elevator = The channel 2 output (elevator or pitch cyclic on Futaba radios), normalized to [-1, 1].
   - Ch.3 Throttle = The channel 3 output (throttle and/or collective on Futaba radios), normalized to [-1, 1].
   - Ch.4 Rudder = The channel 4 output (rudder or tail rotor on Futaba radios), normalized to [-1, 1].
   - Ch.5 Gear = The channel 5 output (usually landing gear on Futaba radios), as a discrete 0 or 1.
   - Ch.6 Flap = The channel 6 output (usually flap on Futaba radios), as a discrete 0 or 1.
   - Ch.7 Aux1 = The channel 7, auxiliary, as a discrete 0 or 1.
   - Ch.8 Aux2 = The channel 8, auxiliary, as a discrete 0 or 1.
4. *Details:* The block is using the same CMEX S-function as the generic *Joystick Interface* block, **sfunjoy.dll**, but it is customizing the outputs and providing only those axes and buttons that are physically available with an 8-channel R/C transmitter. A radio transmitter can be used as a PC joystick only through a special hardware device that samples the PWM signals of the radio and sends them to the PC via USB, serial, or parallel port - see RealFlight Transmitter Interface, or LEW Engineering RCJOY.

5. *Usage:* The block can be used to provide user inputs for control surfaces deflections, throttle setting and other aircraft commands.

4.10 Propulsion

This library folder includes blocks that model various types of aircraft propulsion systems. Although propulsion models can be extremely complex, the implementations in the **AeroSim** library were kept relatively simple and generic, allowing the designer to quickly develop and test a fully-functional aircraft model.

4.10.1 Fixed-Pitch Propeller

The block provides a look-up table fixed-pitch propeller model. It is shown in Fig. 89.

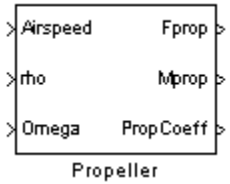


Figure 89: Fixed-Pitch Propeller Block

Block characteristics:

1. *Parameters:*
   - Advance ratio = the advance ratio J vector $n \times 1$.
   - Coefficient of thrust = the coefficient of thrust C_T vector corresponding to the advance ratios above $n \times 1$.
   - Coefficient of power = the coefficient of power C_P vector corresponding to the advance ratios above $n \times 1$.
   - Radius = the propeller radius.
2. *Inputs:*
   - Airspeed = the current airspeed.
   - rho = the air density at current altitude.
   - Omega = the propeller rotation speed (engine shaft rotation speed $\times$ gear ratio), in **rad/s**.
3. *Outputs:*
   - Fprop = the propeller thrust force.
   - Mprop = the propeller torque.
   - PropCoeff = the 3×1 vector of propeller coefficients $[J \quad C_T \quad C_P]^T$.
4. *Details:* The block performs linear 1-D interpolations using $< J, C_T >$ and $< J, C_P >$ look-up tables. The current advance ratio is computed as:

$$J = \frac{\pi V_a}{\Omega R} \tag{65}$$

   where R represents the propeller radius.

   Knowing the thrust and power coefficients we can compute the thrust force and propeller moment:

$$F_p = \frac{4}{\pi^2} \rho R^4 \Omega^2 C_T \tag{66}$$

$$M_p = -\frac{4}{\pi^3} \rho R^5 \Omega^2 C_P \tag{67}$$

5. *Usage:* The block gets its airspeed input from *Wind-axes Velocities* in *Aerodynamics*, the air density from *Standard Atmosphere* in the *Atmosphere* model, and the rotation speed from the *Piston Engine* model in **Propulsion**. The block outputs are used by **Propulsion Force**, and **Propulsion Dynamics** blocks in **Propulsion**. The **Fixed-pitch Propeller** block is used in the **GA Propulsion System** block.

4.10.2 Piston Engine

The *Piston Engine* block provides a simple internal combustion engine model based on look-up tables of engine parameters. The tables needed include 2-dimensional matrix tables for fuel flow and engine power at sea-level, as functions of RPM and manifold pressure (MAP), as well as the RPM and MAP vectors containing the data points for which the above-mentioned tables are given. The block is shown in Fig. 90.

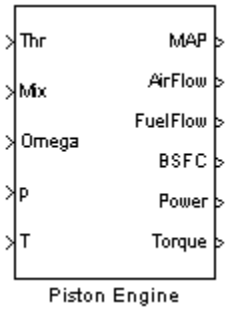


Figure 90: Piston Engine Block

Block characteristics:

1. *Parameters:*
   - RPM vector = the engine speed data points as a $N_{RPM} \times 1$ vector, given in rotations per minute.
   - MAP vector = the manifold pressure data points as a $N_{MAP} \times 1$ vector, given in kPa.
   - Fuel flow look-up table = the mass fuel flow data as a $N_{RPM} \times N_{MAP}$ matrix, given in grams per hour.
   - Power look-up table = the engine power at sea level, as a $N_{RPM} \times N_{MAP}$ matrix, given in Watts.
   - Sea-level pressure = the ambient pressure at which the above engine data was collected, in Pa.
   - Sea-level temperature = the temperature at which the above engine data was collected, in K.
2. *Inputs:*
   - Thr = the throttle fraction (engine control), from 0 to 1.
   - Mix = the air-to-fuel ratio, or mixture.
   - Omega = the current engine shaft rotation speed, in rad/s.
   - p = the atmospheric pressure at current altitude, in Pa.
   - T = the atmospheric temperature at current altitude, in K.
3. *Outputs:*
   - MAP = the manifold air pressure for current throttle setting and altitude, in kPa.
   - Airflow = the instantaneous mass air flow, in kg/s.
   - Fuelflow = the instantaneous mass fuel flow, in kg/s.
   - BSFC = the brake specific fuel consumption, in g/(W*hr).
   - Power = the current engine power, in W.
   - Torque = the torque generated at engine shaft, in N*m.

4. *Details:* The throttle input to this *Piston Engine* block represents the fraction between the manifold pressure and atmospheric pressure. For normally-aspirated engines, the manifold pressure will always be less than or equal to the atmospheric pressure at current altitude, therefore the pressure fraction can take only values from 0 to 1. Practically, the engine runs in a stable manner only for manifold pressures of 50-60 kPa and higher. To be able to provide throttle command as a throttle actuator position, a look-up table that maps pressure ratio to area fraction should be placed ahead of the *Piston Engine* block - this mapping is nonlinear and it depends on the geometry of the intake system.

5. *Usage:* The **Piston Engine** subsystem is used by the **GA Propulsion System** block.

4.10.3 GA Propulsion System

The general-aviation propulsion system includes a fixed-pitch propeller, a piston engine, and the differential equation which is solved for the engine shaft rotation speed. The block is shown in Fig. 91.

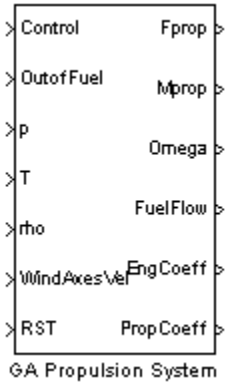


Figure 91: GA Propulsion System Block

Block characteristics:

1. *Parameters:*

   - Initial engine speed = the initial condition for the engine rotation speed integrator, in rad/s.
   - RPM vector = the engine speed data points as a $N_{RPM} \times 1$ vector, given in rotations per minute.
   - MAP vector = the manifold pressure data points as a $N_{MAP} \times 1$ vector, given in kPa.
   - Fuel flow look-up table = the mass fuel flow data as a $N_{RPM} \times N_{MAP}$ matrix, given in grams per hour.
   - Power look-up table = the engine power at sea level, as a $N_{RPM} \times N_{MAP}$ matrix, given in Watts.
   - Sea-level pressure = the ambient pressure at which the above engine data was collected, in Pa.
   - Sea-level temperature = the temperature at which the above engine data was collected, in K.
   - Advance ratio = the advance ratio J vector $n \times 1$.
   - Coefficient of thrust = the coefficient of thrust C_T vector corresponding to the advance ratios above $n \times 1$.
   - Coefficient of power = the coefficient of power C_P vector corresponding to the advance ratios above $n \times 1$.
   - Radius = the propeller radius.
   - Propeller moment of inertia = the moment of inertia of the propeller.
   - Engine moment of inertia = the moment of inertia of the rotating part of the engine (shaft) - generally it is significantly smaller than the propeller moment inertia.

2. *Inputs:*

   - Control = the 3×1 vector of engine control inputs, which are throttle, mixture, and ignition. The ignition can be 1 or 0 to turn the engine on, respectively off.
   - OutofFuel = the "out-of-fuel" has a similar functionality to the ignition switch.

- p = the atmospheric pressure at current altitude, in Pa.
- T = the atmospheric temperature at current altitude, in K.
- rho = the air density at current altitude.
- WindAxesVel = the 3×1 vector of wind axes velocities (airspeed, sideslip angle, angle-of-attack).
- RST = the engine speed integrator reset flag.

3. *Outputs:*
   - Fprop = the 3×1 vector of propulsion forces (the propeller thrust force is on X axis).
   - Mprop = the 3×1 vector of propulsion moments (the torque at engine mount is on X axis).
   - Omega = the engine shaft rotation speed, in rad/s.
   - Fuelflow = the instantaneous mass fuel flow, in kg/s.
   - EngCoeff = the 5×1 vector of engine coefficients, which include MAP, air flow, fuel flow, BSFC, and power.
   - PropCoeff = the 3×1 vector of propeller coefficients $[J \quad C_T \quad C_P]^T$.
4. *Details:* The **GA Propulsion System** includes a **Piston Engine** and a **Fixed-pitch Propeller**. These blocks return the torque provide by the engine, respectively the torque required by the propeller, at current engine shaft rotation speed, current atmospheric conditions, and airspeed. Then the differential equation that describes the dynamics of the propulsion system is:

$$(I_{eng} + I_{prop})\dot{\Omega} = M_{eng} + M_{prop} \tag{68}$$

   This equation is integrated forward in time to compute the engine speed Ω at the next time step.
5. *Usage:* The **GA Propulsion System** is used in the nonlinear aircraft model to compute the propulsion forces and moments that are applied to the vehicle. These will depend on the engine control inputs, atmospheric conditions, altitude, and airspeed.

4.11 Sensors

The **Sensors** library folder includes generic, analog and digital sensor models. These can be placed between the aircraft model and the real or simulated flight control system in order to increase the realism of the hardware or software in the loop simulation. The Sensors library is divided into 4 sub-folders: noise correlations, generic sensors, analog sensors, and digital sensors.

4.11.1 Noise Correlation: Random Walk

The block implements a "Random-Walk" process, by integrating the output of a white-noise source. The block is pictured in Fig. 92.

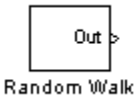


Figure 92: Random Walk Block

Block characteristics:

1. *Parameters:*
   - White-noise seed = the starting sequence of the white-noise source, for example [23341]. Refer to Mathworks documentation on the Simulink block "White-Noise" for more information about the noise seed.
   - Variance = the variance (rms) of the white-noise source.
   - Sample time = the sample time of the white-noise source.
2. *Inputs:*
   - None
3. *Outputs:*
   - Output = a random walk noise signal.
4. *Details:* The random walk process is implemented by integrating the output "w" of a white-noise source:

$$\dot{x} = w \tag{69}$$

5. *Usage:* The random-walk process is used to simulate the noise for sensors whose time correlation characteristic is not known (the random-walk basically assumes infinite correlation time).

4.11.2 Noise Correlation: Gauss-Markov Process

The block implements a Gauss-Markov process with exponentially-decaying correlation. The block is pictured in Fig. 93.

Figure 93: Gauss-Markov Process Block

Block characteristics:

1. *Parameters:*
   - Time constant = the Gauss-Markov time constant, or correlation time.
   - White-noise seed = the starting sequence of the white-noise source, for example [23341]. Refer to Mathworks documentation on the Simulink block "White-Noise" for more information about the noise seed.
   - Variance = the variance (rms) of the white-noise source.
   - Sample time = the sample time of the white-noise source.
2. *Inputs:*
   - None
3. *Outputs:*
   - Output = a noise signal with exponentially-decreasing autocorrelation.
4. *Details:* The Gauss-Markov process is one which has an exponentially-decreasing autocorrelation function. In practical terms, this amounts to the assumption that the process exhibits little correlation between values which are sufficiently well separated in time. If the correlation time is τ and the white-noise input is w then the process dynamics are:
$$\dot{x} = -\frac{1}{\tau}x + w \tag{70}$$
5. *Usage:* The Gauss-Markov process can be used to simulate sensors for which we have an idea about their correlation time. For example, it is reasonable to model GPS output as a Gauss-Markov process with a correlation time of the order of 10^2 seconds.

4.11.3 Simple Sensor - 1st-order dynamics

The block implements a simple sensor that includes saturation limits, time lag (1st-order dynamics) and random walk. The block is shown in Fig. 94.

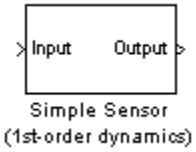


Figure 94: Simple Sensor Block

Block characteristics:

1. *Parameters:*
   - Max value = the upper limit for the saturation block (max sensed value).
   - Min value = the lower limit for the saturation block (min sensed value).
   - Output lag = the time constant τ_s for the sensor dynamics, which are implemented as a transfer function of the form (71).

$$G_s(s) = \frac{1}{\tau_s s + 1} \tag{71}$$

   - Noise seed = the starting sequence of the white-noise source, for example [23341]. Refer to Mathworks documentation on the Simulink block "White-Noise" for more information about the noise seed.
   - Noise variance = the variance (rms) of the white-noise source.
   - Noise sample time = the sample time of the white-noise source.
2. *Inputs:*
   - Input = the measured signal (scalar).
3. *Outputs:*
   - Output = the sensor output signal.
4. *Details:* None.
5. *Usage:* The simple sensor block is used as a basis for creating more detailed sensor models, but it can also be used stand-alone when the sensor characteristics are not known in detail.

4.11.4 Simple Sensor - 2nd-order dynamics

The block implements a simple sensor that includes saturation limits, 2nd-order dynamics, and random walk. The block is shown in Fig. 95.

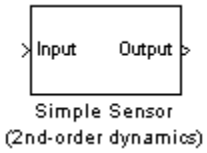


Figure 95: Simple Sensor Block

Block characteristics:

1. *Parameters:*
   - Max value = the upper limit for the saturation block (max sensed value).
   - Min value = the lower limit for the saturation block (min sensed value).
   - Bandwidth = the sensor bandwidth b at a gain $g = -3dB$ and phase $\phi = -90°$. The 2-nd order transfer function is implemented as in (72).

$$G_s(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{72}$$

   where $\omega_n = 2\pi b$ and $\zeta = \frac{1}{2}10^{\frac{-g}{20}}$.
   - Noise seed = the starting sequence of the white-noise source, for example [23341]. Refer to Mathworks documentation on the Simulink block "White-Noise" for more information about the noise seed.
   - Noise variance = the variance (rms) of the white-noise source.
   - Noise sample time = the sample time of the white-noise source.
2. *Inputs:*
   - Input = the measured signal (scalar).
3. *Outputs:*
   - Output = the sensor output signal.
4. *Details:* None.
5. *Usage:* The simple sensor block is used as a basis for creating more detailed sensor models, but it can also be used stand-alone when the sensor characteristics are not known in detail.

4.11.5 Analog Sensor

The block implements a generic analog sensor model that takes a "measured" variable as input (it would normally come from the aircraft dynamic model) and outputs an analog voltage signal which can be provided to an autopilot system. The block is shown in Fig. 96.

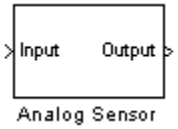


Figure 96: Analog Sensor Block

Block characteristics:

1. *Parameters:*
   - Range = a 1×2 vector with the minimum and maximum values that can be measured by the sensor.
   - Bias point = the zero for the measured signal (0 for most sensors).
   - Bias voltage = the voltage that the sensor will output for the bias point specified above.
   - Scale factor = the conversion factor (gain) from measured signal to voltage.
   - Bandwidth = the sensor bandwidth b at a gain $g = -3dB$ and phase $\phi = -90°$. The 2-nd order transfer function is implemented as in (73).

$$G_s(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{73}$$

   where $\omega_n = 2\pi b$ and $\zeta = \frac{1}{2}10^{\frac{-g}{20}}$.
   - Noise seed = the starting sequence of the white-noise source, for example [23341]. Refer to Mathworks documentation on the Simulink block "White-Noise" for more information about the noise seed.
   - Noise variance = the variance (rms) of the white-noise source.
   - Noise sample time = the sample time of the white-noise source.
2. *Inputs:*
   - Input = the measured signal (scalar).
3. *Outputs:*
   - Output = the analog sensor output, in volts.
4. *Details:* The analog sensor model contains bias, scale-factor conversion, and a Simple Sensor block with 2nd-order dynamics.
5. *Usage:* The block input is taken from the aircraft dynamic model. The output can be provided to an *A/D converter* block for discretization and then to a flight control system model.

4.11.6 A/D Converter

The *A/D Converter* block can discretize an analog input signal given in volts and output a digital signal in discrete counts. The block is pictured in Fig. 97.

Figure 97: A/D Converter Block

Block characteristics:

1. *Parameters:*
   - Voltage range = a 1×2 vector containing the minimum and maximum values of the input voltage.
   - Resolution = the number of bits of resolution, which determine the number of counts available on the specified voltage range. For example, a 10-bit converter will provide $2^{10} = 1024$ counts.
   - Sample time = the time interval at which the analog signal is sampled.
2. *Inputs:*
   - Analog = the analog input signal, in volts.
3. *Outputs:*
   - Digital = the digital signal, in counts.
4. *Details:* The block includes scale-factor conversion from volts to counts, saturation limits on the input signal, rounding to the nearest count value, and zero-order hold on the digital output.
5. *Usage:* The block can be used to digitize analog sensor signals provided by *Analog Sensor* block in *Sensors*. It can also be used individually to model digital sensors (GPS receiver, for example).

4.11.7 Single GPS Measurement

The block models a single GPS measurement with a Gauss-Markov process and a transport delay. The block is shown in Fig. 98.

Figure 98: Single GPS Measurement Block

Block characteristics:

1. *Parameters:*
   - Initial value = the initial value for the block output.
   - Output lag = the transport delay time.
   - Noise correlation time = the Gauss-Markov time constant.
   - Noise seed = the white-noise source starting seed.
   - Noise variance = the white-noise variance.
   - Noise sample time = the white noise sample time.
2. *Inputs:*
   - In = a single GPS measurement (position or velocity).
3. *Outputs:*
   - Out = the GPS measurement output (position or velocity).
4. *Details:* The block includes a Gauss-Markov noise correlation block and a transport delay block.
5. *Usage:* The block is used by the GPS PV model.

4.11.8 GPS PV

The GPS PV (position and velocity) block adds Gauss-Markov processes and transport delay on position and velocity channels. The block is shown in Fig. 99.

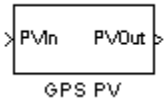


Figure 99: GPS PV Block

Block characteristics:

1. *Parameters:*
   - Initial position = the 3x1 vector of initial position output, in [rad rad m]
   - Initial velocity = the 3x1 vector of initial groundspeed, in m/s.
   - Position lag = the position transport delay, in seconds.
   - Velocity lag = the groundspeed transport delay, in seconds.
   - Horizontal position variance = the white-noise variance for latitude and longitude, in meters.
   - Vertical position variance = the white-noise variance for altitude, in meters.
   - Horizontal velocity variance = the white-noise variance for horizontal groundspeed components, in m/s.
   - Vertical velocity variance = the white-noise variance for vertical velocity, in m/s.
   - Noise correlation time = the Gauss-Markov time corelation of the GPS outputs.
   - Noise sample time = the sample time of the white-noise sources.
2. *Inputs:*
   - PVIn = the 6x1 vector of GPS measurements (position and velocity).
3. *Outputs:*
   - PVOut = the 6x1 vector of GPS measurements (position and velocity).
4. *Details:* The block is using 6 "Single GPS measurement" block for the 3 position and 3 velocity components.
5. *Usage:* Position and velocity inputs to this block should be connected to the sensor output of the aircraft model. The block output can be used for an autopilot or an integrated GPS/INS navigation algorithm.

4.12 Transformations

The **Transformations** library folder includes parameter and reference frame transformations that are required in the aircraft model equations or in autopilot design.

4.12.1 Body-Inertial DCM From Quaternions

The block computes the direction cosine matrix required for transformation from inertial to body frame. It is pictured in Fig. 100.

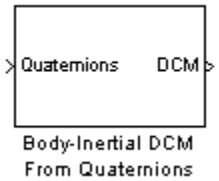


Figure 100: Body-Inertial DCM From Quaternions Block

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - Quaternions = the 4×1 vector of quaternions $\begin{bmatrix} e_0 & e_x & e_y & e_z \end{bmatrix}^T$.
3. *Outputs:*
   - DCM = the 3×3 direction cosine matrix.
4. *Details:* The equation for the direction cosine matrix is straightforward (from [2]):

$$DCM = \begin{bmatrix} e_x^2 + e_0^2 - e_y^2 - e_z^2 & 2(e_x e_y + e_z e_0) & 2(e_x e_z - e_y e_0) \\ 2(e_x e_y - e_z e_0) & e_y^2 + e_0^2 - e_x^2 - e_z^2 & 2(e_y e_z + e_x e_0) \\ 2(e_x e_z + e_y e_0) & 2(e_y e_z - e_x e_0) & e_z^2 + e_0^2 - e_x^2 - e_y^2 \end{bmatrix} \quad (74)$$

5. *Usage:* The block input is taken from *Kinematics (Quaternions)* in the *Equations of Motion*. The block output is used extensively throughout the aircraft model, to transfer vectors from body to inertial frame and vice-versa.

4.12.2 Body-Inertial DCM From Euler Angles

The block computes the direction cosine matrix required for transformation from inertial to body frame. It is pictured in Fig. 101.

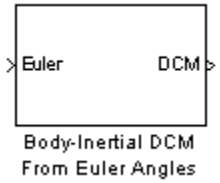


Figure 101: Body-Inertial DCM From Euler Angles Block

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - Euler = the 3×1 vector of Euler angles $[\phi \quad \theta \quad \psi]^T$.
3. *Outputs:*
   - DCM = the 3×3 direction cosine matrix.
4. *Details:* The equation for the direction cosine matrix is:

$$DCM = \begin{bmatrix} C_\theta C_\psi & C_\theta S_\psi & -S_\theta \\ S_\phi S_\theta C_\psi - C_\phi S_\psi & S_\phi S_\theta S_\psi + C_\phi C_\psi & S_\phi C_\theta \\ C_\phi S_\theta C_\psi + S_\phi S_\psi & C_\phi S_\theta S_\psi - S_\phi C_\psi & C_\phi C_\theta \end{bmatrix} \quad (75)$$

5. *Usage:* The block input is taken from *Kinematics (Euler Angles)* in the *Equations of Motion*. The block output is used extensively throughout the aircraft model, to transfer vectors from body to inertial frame and vice-versa.

4.12.3 Body-Wind DCM

The *Body-Wind DCM* block computes the direction cosine matrix for transformation from body to wind frame. The rotation angles for this transformation are the sideslip β and the angle-of-attack α. The block is shown in Fig. 102.

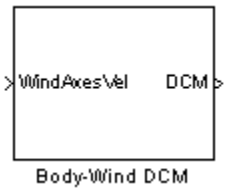


Figure 102: Body-Wind DCM Block

Block characteristics:

1. *Parameters:* none.

2. *Inputs:*

   - WindAxesVel = the 3×1 vector of wind-axes velocity components $[V_a \quad \beta \quad \alpha]^T$, where the angles are provided in **radians**.

3. *Outputs:*

   - DCM = the 3×3 direction cosine matrix.

4. *Details:* The equation for the body-to-wind frame transformation is shown in [3] to be:

$$DCM = \begin{bmatrix} \cos\alpha\cos\beta & \sin\beta & \sin\alpha\cos\beta \\ -\cos\alpha\sin\beta & \cos\beta & -\sin\alpha\sin\beta \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} \tag{76}$$

5. *Usage:* The block is referenced in *Aerodynamic Force* block in the *Aerodynamics* where the aerodynamic forces are transferred from wind axes [Drag Sideforce Lift] to body axes [Fx Fy Fz].

4.12.4 Euler Angles From Quaternions

The block computes the Euler angles from Euler-Rodrigues quaternions. It is shown in Fig. 103.

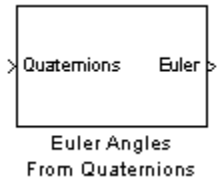


Figure 103: Euler Angles From Quaternions Block

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - Quaternions = the 4×1 vector of quaternions $[e_0 \quad e_x \quad e_y \quad e_z]^T$.
3. *Outputs:*
   - Euler = the 3×1 vector of Euler angles $[\phi \quad \theta \quad \psi]^T$, in radians.
4. *Details:* As shown in [2], the equations for computing the Euler angles are:

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} atan2\left[2(e_0 e_x + e_y e_z), (e_0^2 + e_z^2 - e_x^2 - e_y^2)\right] \\ asin\left[2(e_0 e_y - e_x e_z)\right] \\ atan2\left[2(e_0 e_z + e_x e_y), (e_0^2 + e_x^2 - e_y^2 - e_z^2)\right] \end{bmatrix} \tag{77}$$

5. *Usage:* The block gets its inputs from the *Kinematics (Quaternions)* block in *Equations of Motion*. The resulting Euler angles are only used by the *FS Interface* block in the *Pilot Interface*.

4.12.5 Euler Angles from DCM

The block computes the Euler angles from the direction cosine matrix. It is shown in Fig. 104.

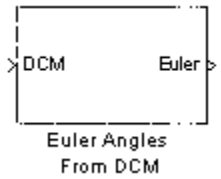


Figure 104: Euler Angles from DCM Block

Block characteristics:

1. *Parameters:* none.

2. *Inputs:*

   - DCM = the 3×3 direction cosine matrix.

3. *Outputs:*

   - Euler = the 3×1 vector of Euler angles $[\phi \quad \theta \quad \psi]^T$, in radians.

4. *Details:* The direction cosine matrix in terms of Euler angles is:

$$DCM = \begin{bmatrix} C_\theta C_\psi & C_\theta S_\psi & -S_\theta \\ S_\phi S_\theta C_\psi - C_\phi S_\psi & S_\phi S_\theta S_\psi + C_\phi C_\psi & S_\phi C_\theta \\ C_\phi S_\theta C_\psi + S_p hi S_\psi & C_\phi S_\theta S_\psi - S_p hi C_\psi & C_\phi C_\theta \end{bmatrix} = \\ = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \tag{78}$$

Then the Euler angles can be extracted as:

$$\theta = asin\,(-C_{13}) \tag{79}$$

$$\phi = asin\left(\frac{C_{23}}{C_\theta}\right) \tag{80}$$

$$\psi = asin\left(\frac{C_{12}}{C_\theta}\right) \tag{81}$$

5. *Usage:* The block is not used anywhere in the aircraft dynamic models built using **AeroSim** library, since the Euler angles can be computed directly from quaternions, using the *Euler Angles* block provided in *Transformations*. The block was provided for completeness and the only situation in which it could be required is in the design of an Integrated Navigation System algorithm.

4.12.6 Quaternions From Euler Angles

The block computes the Euler-Rodrigues quaternions from the Euler angles. It is shown in Fig. 105.

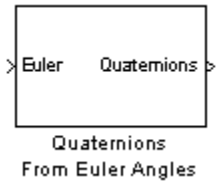


Figure 105: Quaternions From Euler Angles Block

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - Euler = the 3×1 vector of Euler angles $[\phi \quad \theta \quad \psi]^T$, in radians.
3. *Outputs:*
   - Quaternions = the 4×1 vector of quaternions $[e_0 \quad e_x \quad e_y \quad e_z]^T$.
4. *Details:* As shown in [2], the equations for computing the quaternions from Euler angles are:

$$\begin{bmatrix} e_0 \\ e_x \\ e_y \\ e_z \end{bmatrix} = \pm \begin{bmatrix} C_{\phi/2}C_{\theta/2}C_{\psi/2} + S_{\phi/2}S_{\theta/2}S_{\psi/2} \\ S_{\phi/2}C_{\theta/2}C_{\psi/2} - C_{\phi/2}S_{\theta/2}S_{\psi/2} \\ C_{\phi/2}S_{\theta/2}C_{\psi/2} + S_{\phi/2}C_{\theta/2}S_{\psi/2} \\ C_{\phi/2}C_{\theta/2}S_{\psi/2} - S_{\phi/2}S_{\theta/2}C_{\psi/2} \end{bmatrix} \tag{82}$$

5. *Usage:* The block gets its inputs from the *Kinematics (Euler Angles)* block in *Equations of Motion*.

4.12.7 ECEF Position

The block computes the aircraft coordinates in the Earth-Centered Earth-Fixed frame, given the position of the aircraft in the geographic frame. The block is shown in Fig. 106.

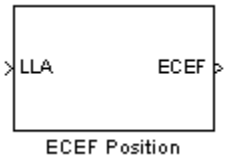


Figure 106: ECEF Position Block

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - LLA = the 3×1 vector of geographic position [Lat Lon Alt] given in **[rad rad m]**.
3. *Outputs:*
   - ECEF = the 3×1 vector of position in ECEF frame [X Y Z], in meters.
4. *Details:* The equations to compute the ECEF position, as presented in [4], are:

$$X_{ECEF} = \left(\frac{r_e}{\sqrt{1 - \varepsilon^2 \sin^2 Lat}} + Alt \right) \cos Lat \cos Lon \quad (83)$$

$$Y_{ECEF} = \left(\frac{r_e}{\sqrt{1 - \varepsilon^2 \sin^2 Lat}} + Alt \right) \cos Lat \sin Lon \quad (84)$$

$$Z_{ECEF} = \left(\frac{r_e(1 - \varepsilon^2)}{\sqrt{1 - \varepsilon^2 \sin^2 Lat}} + Alt \right) \sin Lat \quad (85)$$

5. *Usage:* Although the block is not required in aircraft models built using **AeroSim** library, it can be used in autopilot blocks that implement Great Circle navigation algorithms. The block can get its position input from a GPS receiver or from the *Navigation* block in the *Equations of Motion*.

4.13 Unit Conversion

For consistency with various look-up table models provided in the **AeroSim** library we recommend the use of radians for angular positions, velocities, and accelerations, and the use of metric units for everything else. The **Unit Conversion** blocks were provided for cases when degrees for angles and/or English units for other parameters are required in the aircraft model. In this cases we recommend placing the unit conversion blocks on the inputs and outputs of the aircraft model, such that, internally, the aircraft model still operates in metric units but the model inputs and outputs are provided in the units of choice.

4.13.1 Angular position: Deg 2 rad and Rad 2 deg

The blocks convert angle inputs from degrees to radians and vice-versa. They are shown in Fig. 107.

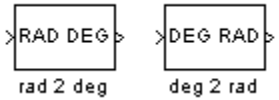


Figure 107: Deg 2 rad and Rad 2 deg Blocks

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - In = angle in degrees, respectively radians.
3. *Outputs:*
   - Out = angle in radians, respectively degrees.
4. *Details:* The conversion factor from degrees to radians is $\pi/180$.
5. *Usage:* All angle inputs for the **AeroSim** library blocks should be given in radians. The blocks can be used by the designer to provide aircraft model inputs/outputs in degrees.

4.13.2 Angular velocity: Rad/s 2 RPM and RPM 2 rad/s

The blocks convert angular velocity inputs from radians-per-second to rotations-per-minute and vice-versa. The blocks are represented in Fig. 108.

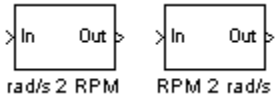


Figure 108: Rad/s 2 RPM and RPM 2 rad/s Blocks

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - In = angular velocity in rad/s, respectively rpm.
3. *Outputs:*
   - Out = angular velocity in rpm, respectively rad/s.
4. *Details:* The conversion factor from rad/s to rpm is $30/\pi$.
5. *Usage:* All angular velocities for the **AeroSim** library blocks should be given in radians-per-second. The blocks can be used by the designer to provide engine/propeller rotation speed in rotations-per-minute.

4.13.3 Distance: ft 2 m and m 2 ft

The blocks convert distances from meters to feet and vice-versa. They are presented in Fig. 109.

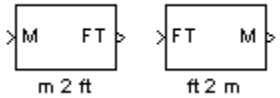


Figure 109: Ft 2 m and m 2 ft Blocks

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - In = distance in meters, respectively feet.
3. *Outputs:*
   - Out = distance in feet, respectively meters.
4. *Details:* The conversion factor from ft to m is 0.3048.
5. *Usage:* Although most blocks in the **AeroSim** library are not influenced by the choice of distance units, there are a few blocks that require metric units. For consistency we recommend use of metric units throughout the aircraft model. Conversion to and from English units can be performed for the model I/O.

4.13.4 Distance: m 2 nm and nm to m

The blocks convert distances from meters to nautic miles and vice-versa. They are presented in Fig. 110.

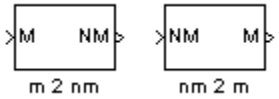


Figure 110: m 2 nm and nm 2 m Blocks

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - In = distance in meters, respectively nautic miles.
3. *Outputs:*
   - Out = distance in nautic miles, respectively meters.
4. *Details:* The conversion factor from nm to m is 1853.24496.
5. *Usage:* Although most blocks in the **AeroSim** library are not influenced by the choice of distance units, there are a few blocks that require metric units. For consistency we recommend use of metric units throughout the aircraft model. Conversion to and from English units can be performed for the model I/O.

4.13.5 Velocity: m/s 2 km/h and km/h 2 m/s

The blocks convert velocities from meters-per-second to kilometers-per-hour and vice-versa. They are presented in Fig. 111.

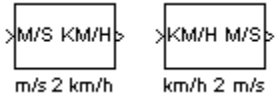


Figure 111: m/s 2 km/h and km/h 2 m/s Blocks

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - In = velocity in m/s, respectively km/h.
3. *Outputs:*
   - Out = velocity in km/h, respectively m/s.
4. *Details:* The conversion factor from m/s to km/h is 3.6.
5. *Usage:* Because velocities are integrated with respect to simulation time (given in seconds), units such as km/h cannot be used for block inputs. For consistency with the distance units, we recommend use of m/s for velocities. The conversion block should be used only for aircraft model I/O.

4.13.6 Velocity: m/s 2 mph and mph 2 m/s

The blocks convert velocities from meters-per-second to miles-per-hour and vice-versa. They are presented in Fig. 112.

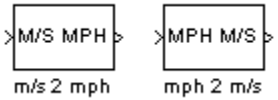


Figure 112: m/s 2 mph and mph 2 m/s Blocks

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - In = velocity in m/s, respectively mph.
3. *Outputs:*
   - Out = velocity in mph, respectively m/s.
4. *Details:* The conversion factor from m/s to mph is 2.2369362921.
5. *Usage:* Because velocities are integrated with respect to simulation time (given in seconds), units such as mph cannot be used for block inputs. For consistency with the distance units, we recommend use of m/s for velocities. The conversion block should be used only for aircraft model I/O.

4.13.7 Velocity: m/s 2 kts and kts 2 m/s

The blocks convert velocities from meters-per-second to knots and vice-versa. They are presented in Fig. 113.

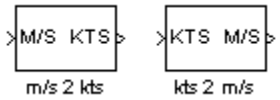


Figure 113: m/s to kts and kts to m/s Blocks

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - In = velocity in m/s, respectively kts.
3. *Outputs:*
   - Out = velocity in kts, respectively m/s.
4. *Details:* The conversion factor from m/s to kts is 1.9425386701.
5. *Usage:* Because velocities are integrated with respect to simulation time (given in seconds), units such as mph cannot be used for block inputs. For consistency with the distance units, we recommend use of m/s for velocities. The conversion block should be used only for aircraft model I/O.

4.13.8 Force: lbf 2 N and N 2 lbf

The block converts a force given in Newtons to pounds of force and vice-versa. It is shown in Fig. 114.

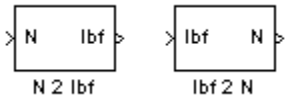


Figure 114: N 2 lbf and lbf 2 N Blocks

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - In = force in N, force in lb respectively.
3. *Outputs:*
   - Out = force in lb, respectively force in N.
4. *Details:* The conversion is: $lb = N \cdot 4.44820070$.
5. *Usage:* User applications.

4.13.9 Mass: lb 2 kg and kg 2 lb

The block converts a mass given in kilograms to pounds of weight and vice-versa. It is shown in Fig. 115.

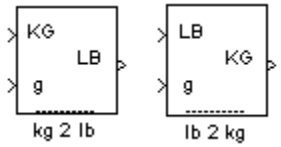


Figure 115: kg 2 lb and lb 2 kg Blocks

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - In = mass in kg, weight in lb respectively.
   - Gravity = the gravitational acceleration at current location, in m/s^2.
3. *Outputs:*
   - Out = weight in lb, respectively mass in kg.
4. *Details:* The conversion is: $lb = kg \cdot g \cdot 0.22481$.
5. *Usage:* The fuel and aircraft masses should be computed in kilograms. Conversion to pounds can be performed for the aircraft model I/O.

4.13.10 Mass: slug 2 kg and kg 2 slug

The block converts a mass given in kilograms to slugs and vice-versa. It is shown in Fig. 116.

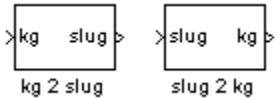


Figure 116: kg 2 slug and slug 2 kg Blocks

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - In = mass in kg, mass in slugs respectively.
3. *Outputs:*
   - Out = mass in slugs, respectively mass in kg.
4. *Details:* The conversion is: $slug = kg \cdot 14.59406605$.
5. *Usage:* The fuel and aircraft masses should be computed in kilograms. Conversion to slugs can be performed for the aircraft model I/O.

4.13.11 Volume: gal 2 l and l 2 gal

The block converts a volume of liquid given in liters to gallons and vice-versa. It is shown in Fig. 117.

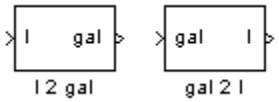


Figure 117: l 2 gal and gal 2 l Blocks

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - In = volume in liters, volume in gallons respectively.
3. *Outputs:*
   - Out = volume in gallons, respectively in liters.
4. *Details:* The conversion is: $gal = l \cdot 3.785412$.
5. *Usage:* Can be used to output current aircraft fuel quantity in English units.

4.13.12 Pressure: Pa 2 in.Hg. and in.Hg. 2 Pa

The blocks convert pressure given in Pascals to inches Hg. and vice-versa. They are shown in Fig. 118.

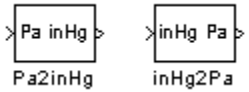


Figure 118: Pa 2 in.Hg. and in.Hg. 2 Pa Blocks

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - In = pressure in Pascals, respectively in in. Hg.
3. *Outputs:*
   - Out = pressure in in.Hg., respectively in Pascals.
4. *Details:* The conversion is: $p_{Hg} = p_{Pa} \cdot 0.00029529$.
5. *Usage:* Can be used for atmospheric pressure conversion. The block is used by the FlightGear-compatible piston-engine model.

4.13.13 Temperature: K 2 F and F 2 K

The blocks convert temperature given in degrees Kelvin to degrees Fahrenheit and vice-versa. They are shown in Fig. 119.

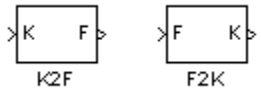


Figure 119: K 2 F and F 2 K Blocks

Block characteristics:

1. *Parameters:* none.
2. *Inputs:*
   - In = temperature in degrees Kelvin, respectively Fahrenheit.
3. *Outputs:*
   - Out = temperature in degrees Fahrenheit, respectively Kelvin.
4. *Details:* The conversion is: $T_F = T_K \cdot 1.8 - 459.67$.
5. *Usage:* The block is used by the piston-engine model in the FlightGear-Compatible library.

4.14 FlightGear-Compatible

AeroSim's FlightGear-compatibility layer includes XML parsers that can load aircraft, engine, and thruster parameters into Matlab structures, as well as the necessary aircraft-dynamics blocks that use such Matlab structures. These aircraft dynamics blocks as well as several complete aircraft models can be found in the **FlightGear-Compatible** library folder and they are presented in detail in this subsection.

4.14.1 Inertia: Empty Aircraft

The *Empty Aircraft* block returns the aircraft mass, CG location and moments of inertia when there is no fuel in the fuel tank(s). The block is shown in Fig. 120.

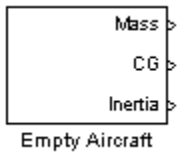


Figure 120: Empty Aircraft Block

Block characteristics:

1. *Parameters:*
   - Aircraft metrics structure = the Matlab structure which contains the parameters provided by the *METRICS* structure of the **JSBSim** XML aircraft configuration file.
2. *Inputs:*
   - none.
3. *Outputs:*
   - Mass = the empty aircraft mass (empty weight).
   - CG = the aircraft CG location $[x_{CG} \quad y_{CG} \quad z_{CG}]$.
   - Inertia = the aircraft moments of inertia $[J_x \quad J_y \quad J_z \quad J_{xz}]$.
4. *Details:* The block simply converts the inertia parameters from the Matlab aircraft structure to Simulink signals.
5. *Usage:* The inertia parameters provided by the *Empty Aircraft* block are summed with inertia parameters of pilot, payloads, and fuel tanks to obtain the gross aircraft inertia.

4.14.2 Inertia: Point Mass

The *Point Mass* provides a convenient way for including additional small components to the total aircraft inertia. The block is shown in Fig. 121.

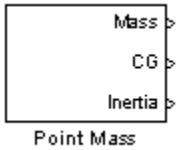


Figure 121: Point Mass Block

Block characteristics:

1. *Parameters:*
   - Point mass structure = a Matlab structure that contains data for a point-mass component (typically it is a sub-structure of the *METRICS* structure.
2. *Inputs:*
   - none.
3. *Outputs:*
   - Mass = the component mass.
   - CG = the component CG location $[x_{CG} \quad y_{CG} \quad z_{CG}]$.
   - Inertia = the component moments of inertia $[J_x \quad J_y \quad J_z \quad J_{xz}]$.
4. *Details:* A consequence of the point-mass assumption is that moments of inertia are zero about its location. Therefore, the only moments of inertia are those created by the component mass with respect to the aircraft reference point. Since moments J_{xy} and J_{yz} are not taken into account, the moment of inertia contribution is only valid if the point-mass components are distributed symmetrically to the aircraft's XZ plane.
5. *Usage:* The inertia parameters provided by the *Point Mass* blocks are summed with inertia parameters of empty airframe and fuel tanks to obtain the gross aircraft inertia.

4.14.3 Propulsion: FG Piston Engine + Fixed-Pitch Prop

The *FG Piston Engine + Fixed-Pitch Prop* block provides a simple but complete propulsion solution for low-speed aircraft models. The block is shown in Fig. 122.

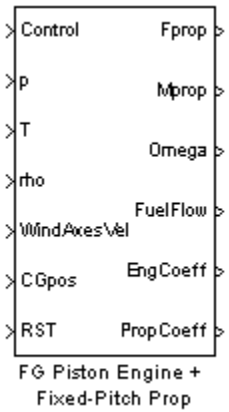


Figure 122: FG Piston Engine + Fixed-Pitch Prop Block

Block characteristics:

1. *Parameters:*
   - Piston engine structure = a Matlab structure which contains the engine data from the JSBSim configuration files.
   - Fixed-pitch prop structure = a Matlab structure which contains the propeller data from the JSBSim configuration files.
   - Sea-level pressure = the sea-level pressure, in Pa.
   - Sea-level temperature = the sea-level temperature, in K.
   - Initial engine speed = the initial value for the engine speed integrator, in rad/s.
2. *Inputs:*
   - Control = the 3×1 vector of propulsion controls (Throttle, Mixture, Ignition).
   - p = the current atmospheric pressure.
   - T = the current outside air temperature.
   - rho = the current air density.
   - WindAxesVel = the 3×1 vector of wind-axes velocities $[Va \quad \beta \quad \alpha]^T$.
   - CGpos = the current aircraft CG location.
   - RST = the integrator reset flag.
3. *Outputs:*
   - Fprop = the propulsion force vector.
   - Mprop = the propulsion moment vector, with respect to the current aircraft CG location.
   - Omega = the current engine rotation speed.
   - FuelFlow = the fuel flow rate.
   - EngCoeff = the engine coefficients (MAP, air flow, fuel flow, BSFC, and power).

- PropCoeff = the propeller coefficients (advance ratio, thrust coefficient, and power coefficient).

4. *Details:* The block includes links to the *FG Piston Engine* and to the *FG Fixed-Pitch Propeller* blocks which can be found in the **AeroSim** library. The block integrates the angular acceleration obtained from the difference between the available engine torque and the required propeller torque, by dividing it by the moment of inertia of the rotating components. Typically the moment of the inertia of the engine shaft is small compared to the propeller moment of inertia, so it can be ignored.

5. *Usage:* The block is used to model a single propulsion unit and its resulting forces and moments. Multi-engine aircraft will use multiple blocks of this type.

4.14.4 Propulsion: FG Piston Engine + Variable-Pitch Prop

The *FG Piston Engine + Variable-Pitch Prop* block provides a more complex propulsion solution for low-speed aircraft models. The block is shown in Fig. 123.

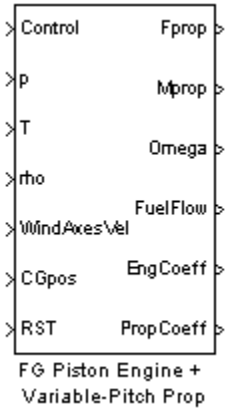


Figure 123: FG Piston Engine + Variable-Pitch Prop

Block characteristics:

1. *Parameters:*
   - Piston engine structure = a Matlab structure which contains the engine data from the JSBSim configuration files.
   - Fixed-pitch prop structure = a Matlab structure which contains the propeller data from the JSBSim configuration files.
   - Sea-level pressure = the sea-level pressure, in Pa.
   - Sea-level temperature = the sea-level temperature, in K.
   - Initial engine speed = the initial value for the engine speed integrator, in rad/s.
2. *Inputs:*
   - Control = the 4×1 vector of propulsion controls (Throttle, Mixture, Ignition, Prop pitch angle).
   - p = the current atmospheric pressure.
   - T = the current outside air temperature.
   - rho = the current air density.
   - WindAxesVel = the 3×1 vector of wind-axes velocities $[Va \quad \beta \quad \alpha]^T$.
   - CGpos = the current aircraft CG location.
   - RST = the integrator reset flag.
3. *Outputs:*
   - Fprop = the propulsion force vector.
   - Mprop = the propulsion moment vector, with respect to the current aircraft CG location.
   - Omega = the current engine rotation speed.
   - FuelFlow = the fuel flow rate.
   - EngCoeff = the engine coefficients (MAP, air flow, fuel flow, BSFC, and power).
   - PropCoeff = the propeller coefficients (advance ratio, thrust coefficient, and power coefficient).

4. *Details:* The block includes links to the *FG Piston Engine* and to the *FG Variable-Pitch Propeller* blocks which can be found in the **AeroSim** library. The block integrates the angular acceleration obtained from the difference between the available engine torque and the required propeller torque, by dividing it by the moment of inertia of the rotating components. Typically the moment of the inertia of the engine shaft is small compared to the propeller moment of inertia, so it can be ignored.

5. *Usage:* The block is used to model a single propulsion unit and its resulting forces and moments. Multi-engine aircraft will use multiple blocks of this type.

4.14.5 Piston Engine: Intake Model

The *Intake Model* computes the manifold presure (MAP) for the current throttle setting and atmospheric pressure. The block is shown in Fig. 124.

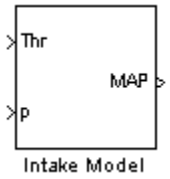


Figure 124: Intake Model Block

Block characteristics:

1. *Parameters:*
   - MAP limits = the min and max values of the MAP.
   - Sea-level ambient pressure = atmospheric pressure at sea-level.
2. *Inputs:*
   - Thr = the current throttle fraction.
   - p = the current atmospheric pressure.
3. *Outputs:*
   - MAP = the current manifold pressure.
4. *Details:* The block performs a linear interpolation of MAP as a function of throttle fraction. The result is then corrected for altitude by multiplying with the ratio of atmospheric pressures at current altitude and at sea-level.
5. *Usage:* The block should be a part of the Piston Engine model and it should provide MAP to the *AirFlow*, *FuelFlow* and *Power* models.

4.14.6 Piston Engine: AirFlow Model

The *Airflow Model* block computes the air flow rate to the engine as a function of MAP, current atmospheric conditions, and engine displacement. The block is shown in Fig. 125.

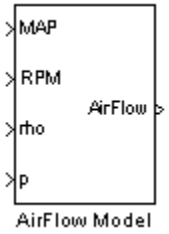


Figure 125: AirFlow Model Block

Block characteristics:

1. *Parameters:*
   - Volumetric efficiency = the volumetric efficiency of the engine (typical value - approximately 0.8).
   - Engine displacement = the displacement of the engine, as specified in the XML engine configuration file.
2. *Inputs:*
   - MAP = the manifold pressure.
   - RPM = the engine rotation speed.
   - rho = the current air density.
   - p = the current atmospheric pressure.
3. *Outputs:*
   - AirFlow = the mass air flow.
4. *Details:* The *AirFlow* block implements the same equations used by the **JSBSim** piston engine model.
5. *Usage:* The block should be a part of the Piston Engine model and it should provide AirFlow to the *FuelFlow* model.

4.14.7 Piston Engine: FuelFlow Model

The *FuelFlow Model* block computes the mass fuel flow to the engine as a function of airflow and current mixture setting. The block is shown in Fig. 126.

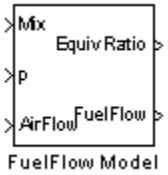


Figure 126: FuelFlow Model Block

Block characteristics:

1. *Parameters:*
   - Sea-level ambient pressure.
2. *Inputs:*
   - Mix = the mixture setting.
   - p = the current atmospheric pressure.
   - AirFlow = the current mass air flow to the engine.
3. *Outputs:*
   - EquivRatio = the equivalence ratio.
   - FuelFlow = the mass fuel flow to the engine.
4. *Details:* The *FuelFlow* block implements the same equations used by the **JSBSim** piston engine model.
5. *Usage:* The block outputs are required by the other engine sub-models such as the computation of engine power and BSFC. The output is also integrated by the *Fuel Tank* model to obtain the current fuel quantity available on-board the aircraft.

4.14.8 Piston Engine: Power Model

The *Power Model* block computes the power released by the engine through the shaft, as a function of manifold pressure and engine shaft rotation speed. The block is shown in Fig. 127.

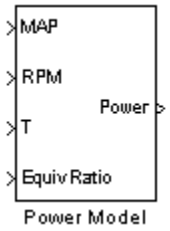


Figure 127: Power Model Block

Block characteristics:

1. *Parameters:*
   - Sea-level ambient temperature = the OAT at sea-level, in degrees K.
   - Maximum power = the maximum engine power at sea-level conditions.
   - Mixture vector = the input argument for the next parameter, which is a look-up table.
   - Power-mixture correlation = the power coefficient look-up table, which is a function of the mixture vector above.
2. *Inputs:*
   - MAP = the current manifold pressure
   - RPM = the current engine rpm
   - T = the current outside-air temperature
   - EquivRatio = the equivalence ratio as computed by the fuel-flow model.
3. *Outputs:*
   - Power = the current engine power output.
4. *Details:* For a more complete explanation of the piston engine model, consult the **JSBSim** manual.
5. *Usage:* The block is used in the *Piston Engine* model. The power output is used to compute the engine shaft torque which is transferred to the propeller. At the same time, the engine torque with changed sign is transferred to the airframe through the engine mount and it represents the moment created by the propulsion system.

4.14.9 Piston Engine: FG Piston Engine

The *FG Piston Engine* combines all of the engine models presented previously in a single component. The block is shown in Fig. 128.

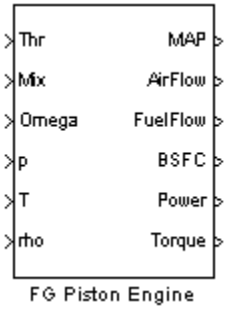


Figure 128: FG Piston Engine Block

Block characteristics:

1. *Parameters:*
   - Engine structure = a Matlab structure which describes the engine parameters
   - Sea-level pressure
   - Sea-level temperature
2. *Inputs:*
   - Thr = the current throttle fraction setting
   - Mix = the current mixture setting
   - Omega = the current engine rotation speed
   - p = the atmospheric pressure
   - T = the outside air temperature
   - rho = the outside air density
3. *Outputs:*
   - MAP = the current manifold pressure
   - AirFlow = the current mass air flow
   - FuelFlow = the current mass fuel flow
   - BSFC = the brake-specific fuel consumption
   - Power = the current power output
   - Torque = the current torque output
4. *Details:* For a more complete explanation of the piston engine model, consult the **JSBSim** manual.
5. *Usage:* The *FG Piston Engine* model along with a propeller form a general-aviation propulsion system. The block is referenced in *FG Piston Engine + Fixed-Pitch Prop* and in *FG Piston Engine + Variable-Pitch Prop*.

4.14.10 Propeller Thruster: FG Fixed-Pitch Propeller

The *FG Fixed-Pitch Propeller* provides a simple propeller model with look-up tables for coefficients of power and thrust as functions of advance ratio. The block is shown in Fig. 129.

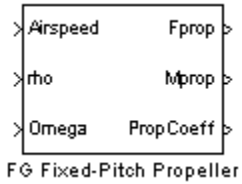


Figure 129: FG Fixed-Pitch Propeller Block

Block characteristics:

1. *Parameters:*
   - Thruster structure = a Matlab structure which contains the propeller parameters.
2. *Inputs:*
   - Airspeed = the current true-airspeed
   - rho = the current atmospheric density
   - Omega = the current propeller rotation speed.
3. *Outputs:*
   - Fprop = the propeller thrust force
   - Mprop = the propeller resisting torque
   - PropCoeff = the propeller coefficient vector $[J \quad C_T \quad C_P]$
4. *Details:* Consult the **AeroSim** *Fixed-pitch Propeller* block reference for more details regarding the propeller model equations.
5. *Usage:* The block is used in the *FG Piston Engine + Fixed-Pitch Prop* general-aviation propulsion system model.

4.14.11 Propeller Thruster: FG Variable-Pitch Propeller

The *FG Variable-Pitch Propeller* provides a simple propeller model with look-up tables for coefficients of power and thrust as functions of advance ratio and blade pitch angle. The block is shown in Fig. 130.

Figure 130: FG Variable-Pitch Propeller

Block characteristics:

1. *Parameters:*
   - Thruster structure = a Matlab structure which contains the propeller parameters.
2. *Inputs:*
   - Airspeed = the current true-airspeed.
   - rho = the current atmospheric density.
   - Omega = the current propeller rotation speed.
   - Pitch = the current propeller pitch setting.
3. *Outputs:*
   - Fprop = the propeller thrust force.
   - Mprop = the propeller resisting torque.
   - PropCoeff = the propeller coefficient vector. $[J \quad C_T \quad C_P]$.
4. *Details:* The model is a more general formulation for the propeller model presented previously. The look-up tables $C_T = C_T(J)$ and $C_P = C_P(J)$ are now bi-dimensional, the second input argument being the blade pitch angle.
5. *Usage:* The block is used in the *FG Piston Engine + Variable-Pitch Prop* general-aviation propulsion system model.

4.14.12 Tank: Fuel Tank

The *Fuel Tank* model integrates the mass fuel flow to obtain the fuel tank inertia parameters at each time step. The block is shown in Fig. 131.

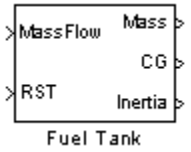


Figure 131: Fuel Tank Block

Block characteristics:

1. *Parameters:*
   - Initial mass = the initial value for the fuel flow integrator.
   - Tank structure = a Matlab structure which contains the tank parameters read from the **JSBSim** configuration file.
2. *Inputs:*
   - MassFlow = the mass fuel flow out of the tank (use negative input if the fuel flows into the tank).
   - RST = the integrator reset flag (reset performed on rising edge).
3. *Outputs:*
   - Mass = the current mass of the fuel in the tank.
   - Inertia = the current moments of inertia of the fuel tank, in aircraft coordinates.
4. *Details:* The fuel tank model assumes a spherical shape for computing the moments of inertia.
5. *Usage:* The block is used in the aircraft inertia model to provide the time-varying inertia characteristics due to fuel consumption.

4.14.13 Aerodynamics: Value

The *Value* block models an aerodynamic coefficient as a scalar gain. The block is shown in Fig. 132.

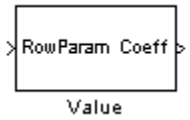


Figure 132: Value Block

Block characteristics:

1. *Parameters:*
   - Coefficient = the scalar coefficient
2. *Inputs:*
   - RowParam = the scalar input parameter
3. *Outputs:*
   - Coeff = the scalar aerodynamic coefficient
4. *Details:* The block provides a method for computing aerodynamic coefficients which depend on only a single parameter.
5. *Usage:* The block can be used to model aerodynamic coefficient build-up using linear parameters (stability derivatives). For example, to account for the roll moment due to yaw rate we can use the value block to compute the roll moment contribution $C_l^r \cdot r$.

4.14.14 Aerodynamics: Vector

The *Vector* block returns an aerodynamic coefficient by performing a table look-up on a given flight parameter. The block is shown in Fig. 133.

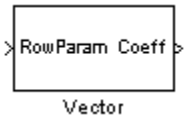


Figure 133: Vector Block

Block characteristics:

1. *Parameters:*
   - Coefficient = the 1-D look-up table for the aerodynamic coefficient
2. *Inputs:*
   - RowParam = the scalar input parameter
3. *Outputs:*
   - Coeff = the scalar aerodynamic coefficient
4. *Details:* The block performs an interpolation on a 1-D look-up table.
5. *Usage:* The block can be used to model aerodynamic coefficients that depend on a single parameter. For example, the induced drag can be modeled as $CD_i = CD_i(C_L)$ using the *Vector* block.

4.14.15 Aerodynamics: Table

The *Table* block computes an aerodynamic coefficient by interpolation through a 2-D look-up table, as function of two flight parameters. The block is shown in Fig. 134.

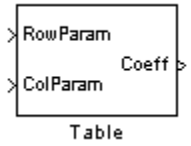


Figure 134: Table Block

Block characteristics:

1. *Parameters:*
   - Coefficient = the 2-D look-up table for the aerodynamic coefficient
2. *Inputs:*
   - RowParam = the input parameter corresponding to the row of the look-up table.
   - ColParam = the input parameter corresponding to the column of the look-up table.
3. *Outputs:*
   - Coeff = the aerodynamic coefficient
4. *Details:* The block performs an interpolation on a 2-D look-up table.
5. *Usage:* The block can be used to model aerodynamic coefficients that depend on more than one parameter. For example, the block can be used to model the lift coefficient which depends on angle-of-attack and flap setting $C_L = C_L(\alpha, \delta_f)$.

4.14.16 Aerodynamics: Coefficient

The *Coefficient* represents a configurable subsystem that allows the user to choose one of the following types: *Value*, *Vector*, or *Table*. The block is shown in Fig. 135.

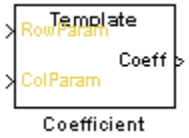


Figure 135: Coefficient Block

Block characteristics:

1. *Parameters:*
   - Coefficient = the aerodynamic coefficient structure, corresponding to an aerodynamic coefficient from the **JSBSim** configuration file. It can be a scalar value, a 1-D or a 2-D look-up table.
2. *Inputs:*
   - RowParam = the first input parameter, it is the only input in the case of a *Value* or a *Vector* coefficient; it is the row argument for a *Table*.
   - ColParam = the second input parameter is required only of the coefficient block is configured as a *Table*, in which case it represents the column argument.
3. *Outputs:*
   - Coeff = the aerodynamic coefficient.
4. *Details:* The block can be configured as one of the aerodynamic coefficients presented previously.
5. *Usage:* The block is used in the aerodynamic coefficient build-up.

4.14.17 Complete Aircraft: Cessna-172

The block contains a pre-built model of the single-engine **C-172** with aircraft parameters provided by the **JSBSim c172.xml** configuration file. The block is shown in Fig. 136.

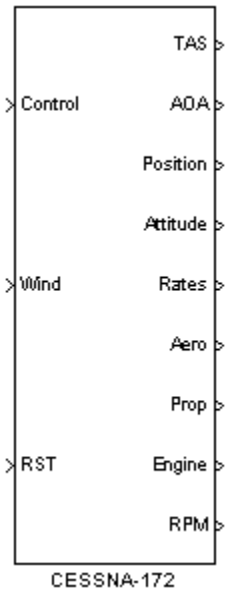


Figure 136: Cessna-172 Block

Block characteristics:

1. *Parameters:*

   - Aircraft name = a string which contains the name of the JSBSim aircraft model (in this case it is c172).
   - FlightGear path = the main path for the FlightGear installation (used for locating the aircraft configuration file).
   - Initial velocities = the initial ground speed components, in geodetic frame $[V_N \quad V_E \quad V_D]^T$.
   - Initial angular rates = the initial aircraft angular rates $[p \quad q \quad r]^T$.
   - Initial quaternions = the initial aircraft attitude, as a quaternion representation $[e_0 \quad e_x \quad e_y \quad e_z]^T$.
   - Initial position = the initial location of the aircraft, in geographic coordinates $[Lat \quad Lon \quad h]^T$.
   - Initial engine speed = the initial engine rotation speed, in rad/s.
   - Initial fuel = a row vector with the initial fuel mass in the left and right fuel tanks.
   - Ground altitude = the ground altitude above sea-level, at current location.
   - Sample time = the simulation sample time, in seconds.

2. *Inputs:*

   - Control = the aircraft control vector, which includes flap, elevator, aileron, rudder, throttle, mixture, and ignition.
   - Wind = the background wind velocity components in geodetic frame $[W_N \quad W_E \quad W_D]$.
   - RST = the integrator reset flag. All integrators reset on the rising edge.

3. *Outputs:*

   - TAS = the true airspeed, in m/s.
   - AOA = the angle-of-attack, in rad.
   - Position = the aircraft position in geographic frame. Latitude and longitude are given in rad, altitude in m.
   - Attitude = the aircraft attitude given in Euler angles, in rad.
   - Rates = the angular rates in body axes, in rad/s.
   - Aero = the aerodynamic coefficients.
   - Prop = the propeller coefficients.
   - Eng = the piston engine coefficients.
   - RPM = the engine rotation speed in rot/min.

4. *Details:* The Simulink model is built using components from the *FlightGear-Compatible* library folder. The model matches the aircraft configuration specified in the **JSBSim** XML configuration file **c172.xml**. This includes a single engine with fixed-pitch prop, and two wing-mounted fuel tanks. The model does not include ground roll/contact dynamics.

5. *Usage:* The complete aircraft model can be used for modeling and simulating the dynamics of a Cessna-172. Control inputs can be provided from a joystick for manual flight, or from an autopilot block for autonomous flight. Simulation output can be visualized using the Flight Simulator or FlightGear interface blocks.

4.14.18 Complete Aircraft: Cessna-182

The block contains a pre-built model of the single-engine **C-182** with aircraft parameters provided by the **JSBSim c182.xml** configuration file. The block is shown in Fig. 137.

Figure 137: Cessna-182 Block

Block characteristics:

1. *Parameters:*

   - Aircraft name = a string which contains the name of the JSBSim aircraft model (in this case it is c172).
   - FlightGear path = the main path for the FlightGear installation (used for locating the aircraft configuration file).
   - Initial velocities = the initial ground speed components, in geodetic frame $[V_N \quad V_E \quad V_D]^T$.
   - Initial angular rates = the initial aircraft angular rates $[p \quad q \quad r]^T$.
   - Initial quaternions = the initial aircraft attitude, as a quaternion representation $[e_0 \quad e_x \quad e_y \quad e_z]^T$.
   - Initial position = the initial location of the aircraft, in geographic coordinates $[Lat \quad Lon \quad h]^T$.
   - Initial engine speed = the initial engine rotation speed, in rad/s.
   - Initial fuel = a row vector with the initial fuel mass in the left and right fuel tanks.
   - Ground altitude = the ground altitude above sea-level, at current location.
   - Sample time = the simulation sample time, in seconds.

2. *Inputs:*

   - Control = the aircraft control vector, which includes flap, elevator, aileron, rudder, throttle, mixture, ignition, and prop blade pitch angle.
   - Wind = the background wind velocity components in geodetic frame $[W_N \quad W_E \quad W_D]$.
   - RST = the integrator reset flag. All integrators reset on the rising edge.

3. *Outputs:*

   - TAS = the true airspeed, in m/s.
   - AOA = the angle-of-attack, in rad.
   - Position = the aircraft position in geographic frame. Latitude and longitude are given in rad, altitude in m.
   - Attitude = the aircraft attitude given in Euler angles, in rad.
   - Rates = the angular rates in body axes, in rad/s.
   - Aero = the aerodynamic coefficients.
   - Prop = the propeller coefficients.
   - Eng = the piston engine coefficients.
   - RPM = the engine rotation speed in rot/min.

4. *Details:* The Simulink model is built using components from the *FlightGear-Compatible* library folder. The model matches the aircraft configuration specified in the **JSBSim** XML configuration file **c182.xml**. This includes a single engine with variable-pitch prop, and two wing-mounted fuel tanks. The model does not include ground roll/contact dynamics.

5. *Usage:* The complete aircraft model can be used for modeling and simulating the dynamics of a Cessna-182. Control inputs can be provided from a joystick for manual flight, or from an autopilot block for autonomous flight. Simulation output can be visualized using the Flight Simulator or FlightGear interface blocks.

4.14.19 Complete Aircraft: Cessna-310

The block contains a pre-built model of the twin-engine **C-310** with aircraft parameters provided by the **JSBSim c310.xml** configuration file. The block is shown in Fig. 138.

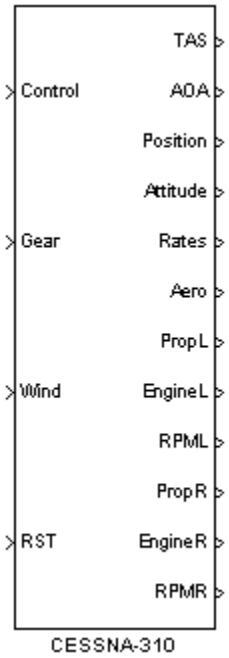


Figure 138: Cessna-310 Block

Block characteristics:

1. *Parameters:*

- Aircraft name = a string which contains the name of the JSBSim aircraft model (in this case it is c172).
- FlightGear path = the main path for the FlightGear installation (used for locating the aircraft configuration file).
- Initial velocities = the initial ground speed components, in geodetic frame $[V_N \quad V_E \quad V_D]^T$.
- Initial angular rates = the initial aircraft angular rates $[p \quad q \quad r]^T$.
- Initial quaternions = the initial aircraft attitude, as a quaternion representation $[e_0 \quad e_x \quad e_y \quad e_z]^T$.
- Initial position = the initial location of the aircraft, in geographic coordinates $[Lat \quad Lon \quad h]^T$.
- Initial engine speed = the initial engine rotation speed, in rad/s.
- Initial fuel = a row vector with the initial fuel mass in the left and right fuel tanks (4 tanks).
- Ground altitude = the ground altitude above sea-level, at current location.
- Sample time = the simulation sample time, in seconds.

2. *Inputs:*

- Control = the aircraft control vector, which includes flap, elevator, aileron, rudder, left engine controls (throttle, mixture, ignition, and prop blade pitch angle) and right engine controls (similar).

- Gear = the retractable landing gear position, used for computing aerodynamic coefficients (1 = gear down, 0 = gear up).
- Wind = the background wind velocity components in geodetic frame $[W_N \quad W_E \quad W_D]$.
- RST = the integrator reset flag. All integrators reset on the rising edge.

3. *Outputs:*
   - TAS = the true airspeed, in m/s.
   - AOA = the angle-of-attack, in rad.
   - Position = the aircraft position in geographic frame. Latitude and longitude are given in rad, altitude in m.
   - Attitude = the aircraft attitude given in Euler angles, in rad.
   - Rates = the angular rates in body axes, in rad/s.
   - Aero = the aerodynamic coefficients.
   - PropL = the left propeller coefficients.
   - EngL = the left piston engine coefficients.
   - RPML = the left engine rotation speed in rot/min.
   - PropR = the right propeller coefficients.
   - EngR = the right piston engine coefficients.
   - RPMR = the right engine rotation speed in rot/min.
4. *Details:* The Simulink model is built using components from the *FlightGear-Compatible* library folder. The model matches the aircraft configuration specified in the **JSBSim** XML configuration file **c310.xml**. This includes two engines with variable-pitch props, and four wing-mounted fuel tanks. The model does not include ground roll/contact dynamics.
5. *Usage:* The complete aircraft model can be used for modeling and simulating the dynamics of a Cessna-310. Control inputs can be provided from a joystick for manual flight, or from an autopilot block for autonomous flight. Simulation output can be visualized using the Flight Simulator or FlightGear interface blocks.

References

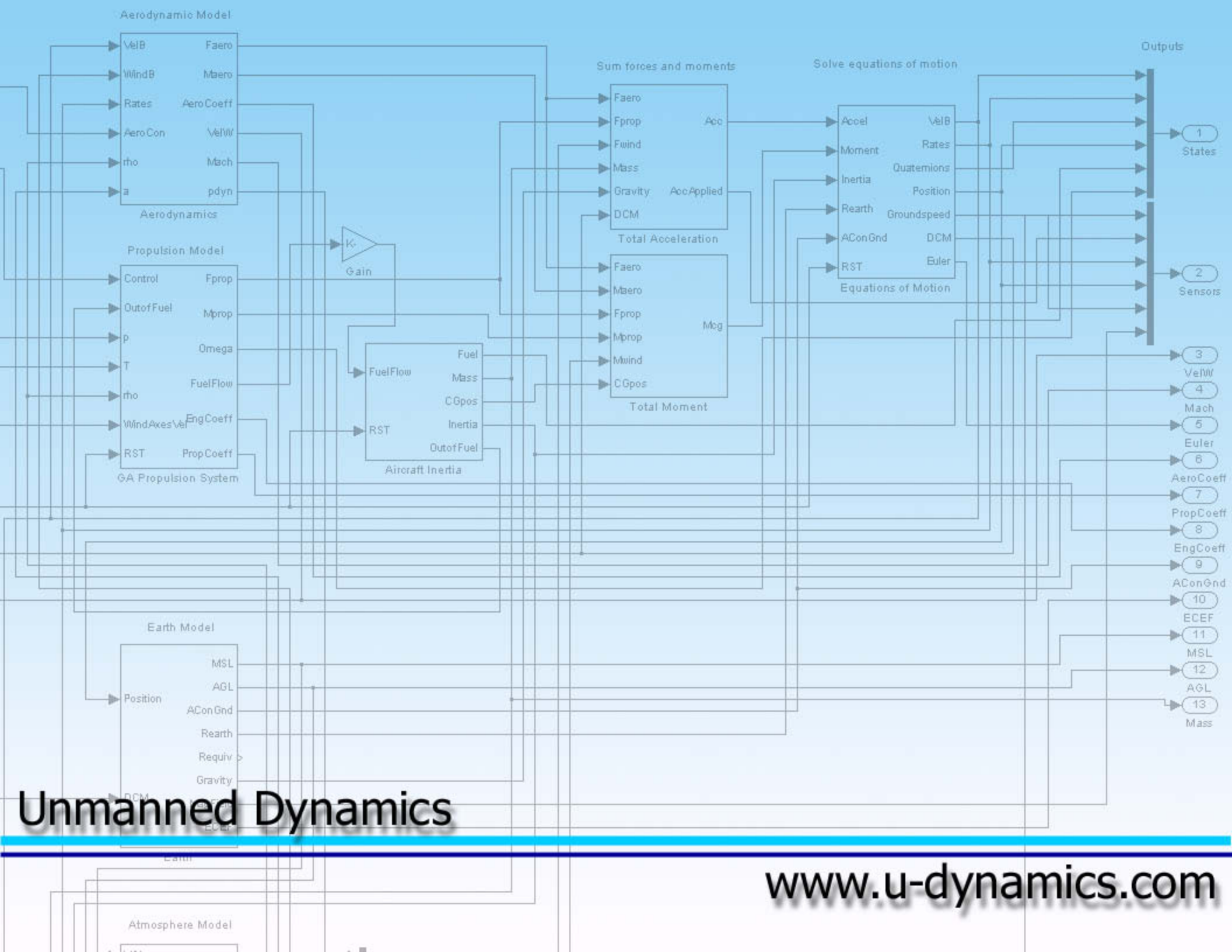
[1] Farrell J. A. and Barth M. *The Global Positioning System & Inertial Navigation*. McGraw Hill, 1999. ISBN 0-07-022045-X.

[2] Phillips W. F. Hailey C. E. and Gebert G. A. Review of Attitude Representations Used for Aircraft Kinematics. *Journal of Aircraft*, 38:718–737, 2001.

[3] Stevens B. L. and Lewis F. L. *Aircraft Control and Simulation*. John Wiley & Sons, Inc., 1992. ISBN 0-471-61397-5.

[4] Rogers R. M. *Applied Mathematics in Integrated Navigation Systems*. American Institute of Aeronautics and Astronautics, Inc., 2000. ISBN 1-56347-397-6.

[5] Smetana F. O. *Evaluation of Stability Derivatives*.



Unmanned Dynamics

www.u-dynamics.com