



Piccolo Simulator

December 3, 2009

Author:

Bill Vaglianti / Marius Niculescu / Jeff Hammitt

2621 Wasco Street / PO Box 1500 / Hood River, OR 97031

(541) 387-2120 phone / (541) 387-2030 fax

www.cloudcaptech.com / sales.cct@goodrich.com / support.cct@goodrich.com



Cloud Cap Technology, a Goodrich Company

Table of Contents

1	Introduction.....	4
2	Dynamics Model Overview	5
3	Creating Simulator Model Overview	6
4	Aerodynamic Model	6
4.1	AVLEditor Commands	6
4.2	Aircraft Editor	7
4.3	Surface Editor	8
4.3.1	New Section	10
4.3.2	Control Surfaces.....	11
4.3.3	Geometric Translation	13
4.3.4	Vortex Lattice	14
4.4	Body Editor	15
4.5	XFOIL Analysis.....	16
4.6	AVL Analysis	18
4.7	Control Surfaces.....	19
5	Inertia Model.....	20
6	Propulsion Model.....	21
6.1	Engine Models	22
6.1.1	Piston Engine Model.....	22
6.1.2	Electric Motor Model.....	23
6.2	Engine Actuator Models	24
6.2.1	Fixed-Pitch Propeller Model.....	25
6.2.2	Simple Rigid Rotor	27
6.2.3	Helicopter Main Rotor	28
6.2.4	Helicopter Tail Rotor	33
7	Landing Gear Model	35
8	Sensor Models.....	38
9	Actuator Models.....	40
10	Launcher Model	41
11	Initialization	42

12	Piccolo Parameters.....	42
13	Installation and Operation.....	43
13.1	Installing FlightGear	44
13.1.1	FlightGear Version 0.9.2.....	44
13.1.2	FlightGear Version 0.9.4 / 0.9.8 / 0.9.9 / 0.9.10	44
13.2	Starting Simulator	44
13.3	FlightGear View Options.....	45
13.4	Displaying Multiple Aircraft.....	47
13.5	Displaying Custom 3-D Aircraft Models.....	48
14	Appendix.....	49
14.1	Inertia Data.....	49
14.2	Stability Derivative List.....	49
14.3	Prop Example File.....	50

1 Introduction

Piccolo is an avionics system for flying small unmanned aircraft. A cornerstone of Piccolo's development environment is the hardware-in-the-loop (HiL) and software-in-the-loop (SiL) simulation system. The Simulator allows the aircraft control laws and mission functionality to be tested without risking hardware in flight test. Although HiL/SiL simulation can not replace flight testing, it measurably reduces the likelihood of failure by detecting bugs and deficiencies in the lab. To facilitate this function, the Piccolo Developers Kit includes a Simulator that can be used to test the performance of a Piccolo implementation.

Hardware-in-the-Loop

In HiL mode the Simulator uses an external CAN interface to connect to an avionics. Piccolo sends servo control information over the CAN bus, and accepts external sensor data on the CAN bus. The Simulator runs on a PC with a CAN interface card and closes the loop by reading the actuator positions, applying them to an aircraft dynamics model, calculating new sensor data, and putting that data on the CAN bus.

FlightGear

In order to visualize the state of the aircraft the Simulator sends UDP network packets to another PC running the open source FlightGear Flight Simulator (FGFS) program. FlightGear accepts the simulation state packets in favor of its own dynamics model, when properly configured for this. This provides an attractive graphics interface for visualizing the performance of the aircraft.

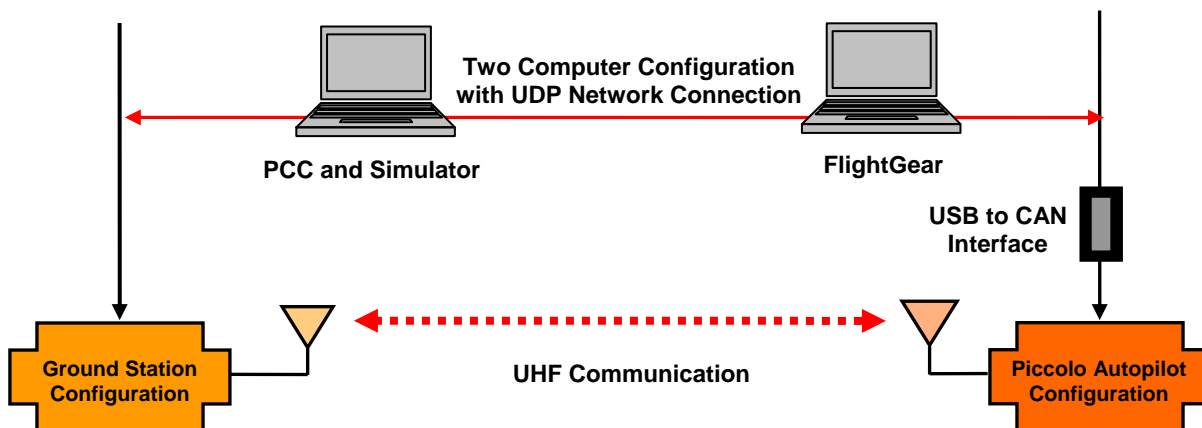


Figure 1 - Hardware-in-the-Loop (HiL)

Software-in-the-Loop

SiL is largely the same except the avionics is virtualized as an application on the PC (PiccoloPC.exe) and the Simulator connects to that application over a TCP socket interface. This is useful for cases where a full hardware laboratory is not available.

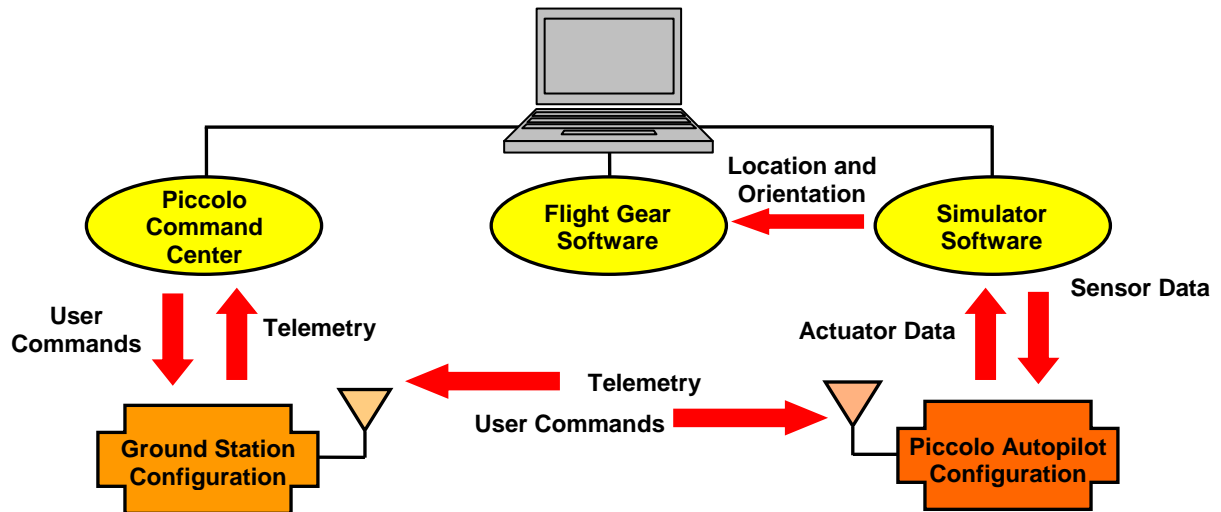


Figure 2 - Software-in-the-Loop

2 Dynamics Model Overview

The heart of the Simulator is the dynamics model¹. It is the sum of all the components that affect the dynamics and kinematics of the aircraft, including: aerodynamics, propulsion, ground contact forces, and inertia effects. The dynamics model contains an integration method that estimates the state of the aircraft based on the previous state and new control surface signals.

There are seven main components of the dynamics model. Each of these components are defined or referenced in the main Simulator input text file.

- Aerodynamics Model
- Propulsion Model
- Sensor Model
- Actuator Model
- Launcher Model.
- Inertia Model
- Landing Gear Model

¹The Simulator is built with a modular architecture that allows third party force and moment models to be plugged-in. This document describes the standard dynamics model that Cloud Cap Technology provides.

3 Creating Simulator Model Overview

The following are the basic steps for developing an aircraft model for the Simulator:

1. Measure aircraft geometry, pull data from 3-view / solid model
2. Determine CG location
3. Create AVL model
4. Refine AVL model with XFOIL data
5. Generate XML aerodynamics model file
6. Model aircraft inertia data
7. Create prop model
8. Create Piccolo Simulator model template
9. Set up Piccolo autopilot configuration (control surfaces, tail configuration, aircraft parameters)
10. Test with software-in-loop simulation and FlightGear visualization
11. Verify manual control response, required control surface limits
12. Generate autopilot gains
13. Load the aircraft file into the Simulator
14. The model is now ready for simulation

4 Aerodynamic Model

To create the aerodynamic model for the Piccolo Simulator, CCT leverages off a program called AVL. AVL is a vortex lattice code developed by Prof. Drela of MIT that models linear aerodynamics with a few nonlinear extras. It is useful for modeling unusual aircraft configurations. So far Cloud Cap has used it to successfully model 20 aircraft. As with all aerodynamic models, it is critical to compare models results with real world dynamics. One of the draw backs of the AVL program is the text based interface. AVLEditor is a program developed by Cloud Cap to visually simplify the modeling process by using a graphical user interface (GUI).

Standard AVL models are generated using a keyword text based data input. AVLEditor replaces this text based data input with a GUI based data entry system that generates a properly formatted AVL data file for you. Cloud Cap modified the stock AVL code to output an XML file of the stability derivatives representing a virtual wind tunnel angle of attack sweep. This XML file is then used for the aerodynamics model in the Simulator. For more detailed information on the specifics of AVL please reference the [AVL documentation](#).

4.1 AVLEditor Commands

AVLEditor breaks the aerodynamic modeling process down into four major groups: Aircraft data, Surface data, Body data, and Airfoil data. These options can be accessed from the **Model** menu in the **AVL Model Editor** window or through the icon menu bar.

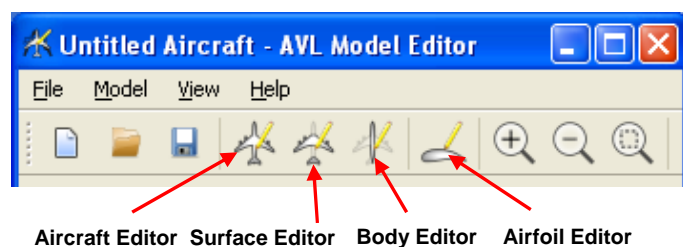


Figure 3 - AVL Editor Commands

The following depicts the various ways of navigating in the **AVL Model Editor** window.

- **Pan:** Left click drag, direction keyboard arrows
- **Rotate:** Right click drag, Ctrl + directional keyboard arrow
- **Zoom:** GUI tool, mouse wheel, right click → zoom in/out

4.2 Aircraft Editor

This section is used to define the basic aircraft parameters used by AVL. The CG is used as the reference point by which the moment and rotation rates are defined. Typically, the CG point is empirically measured from the aircraft being modeled. If no CG is specified, then it is roughly estimated by using an approximation from the body and surfaces. The following reference dimensions are used for calculation of the output aerodynamic coefficients:

- S is the reference area used to define all the coefficients.
- C is the reference chord used to demine the pitching moment.
- B is the reference span used to define roll and yaw moments.

Cruise velocity is used to determine the Mach number in AVL and the Reynolds numbers in XFOIL. AVL treats compressibility using the Prandtl-Glauert transformation. This transformation is a function of Mach number, and is valid to Mach numbers of ~0.6-0.7. For swept wings, the PG model should be judged using the wing-perpendicular Mach number. XFOIL uses the cruise velocity and local chord lengths to calculate the Reynolds numbers for Drag polar determination. The default profile drag value, CD_p , is added to the geometry and applied to the CG.

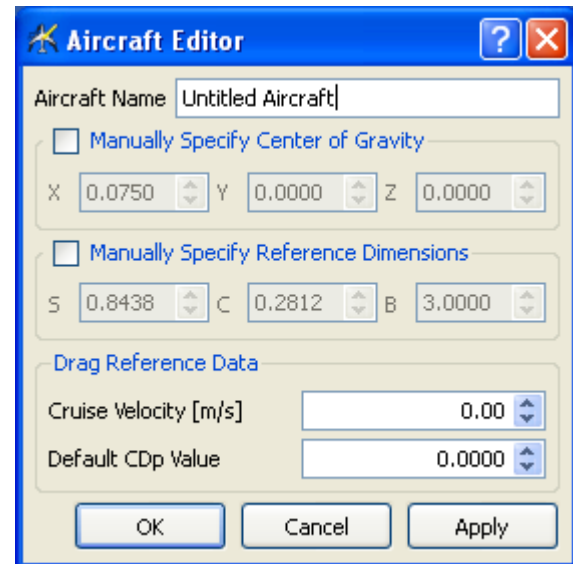


Figure 4 - Aircraft Editor

4.3 Surface Editor

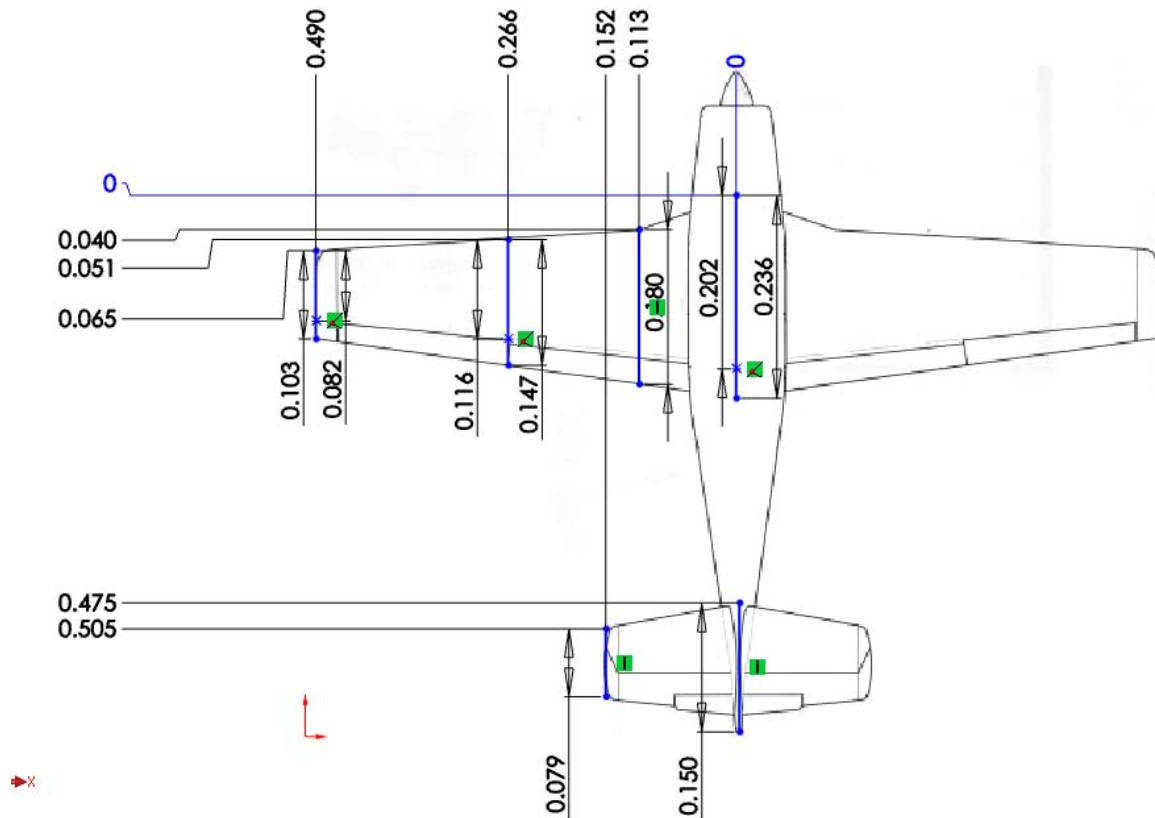


Figure 5 - Aircraft Surface Model

The first step in developing a model is to create the surfaces of the aircraft. This is done using the **Surface Editor** tool. Surfaces are created using sections. Each section contains position, chord, incidence and various other parameters. This allows you to create a wing with varying chord, sweep, and di/anedral. Sections also are used to define the beginning and ends of control surfaces.

The main interface to the surface editor is in the lower left corner of the window. This allows you to create new surfaces, add/delete sections to surface, or add/delete control surfaces on the surface. After any type of modification, it is important to click the **Apply** button for the changes to be saved.

To create a new surface, go to **Model » Surface Editor** or click the **Surface Editor** icon. Click the **New Surface** button and enter the surface name. In this example, we will create a basic wing. AVL creates a basic wing made of two sections symmetric about the 2nd section. To simplify the modeling process AVLEditor can model symmetric surfaces.

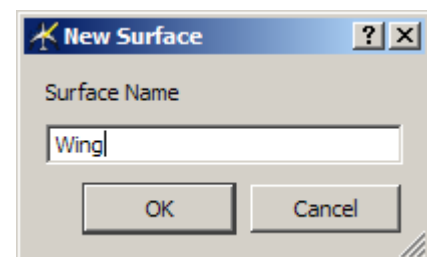


Figure 6 - New Surface

To edit the size or shape of the wing, use the surface editor tool. Positioning the cursor over each section highlights the section in the **AVL Model Editor** window. When using the Y-symmetric feature, both sections are highlighted.

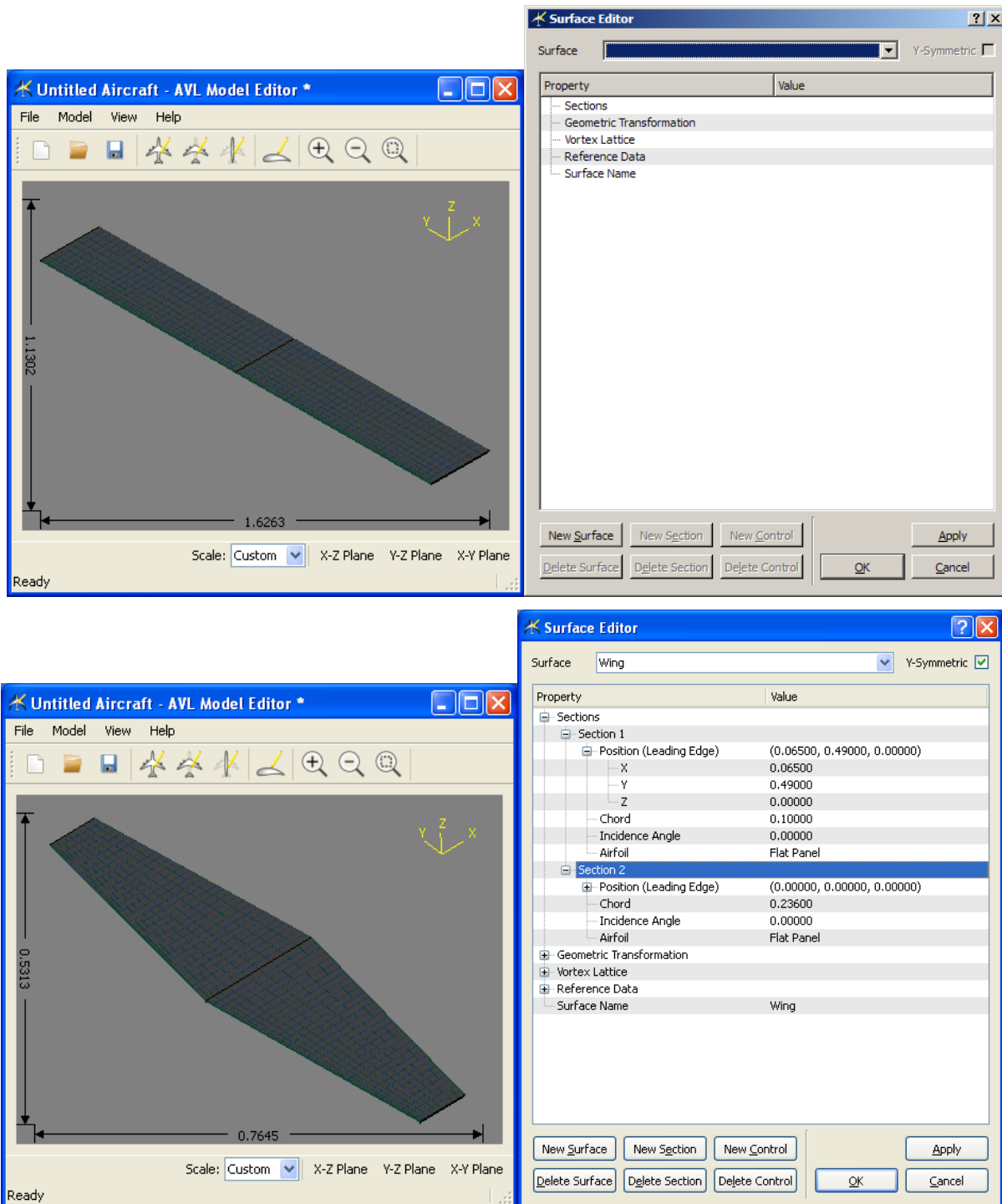


Figure 7 - Editing a Surface

4.3.1 New Section

Use the **New Section** tool to define a new section in a surface. When inserting a new section, the tool will place a new section in between the selected section and the section after.

You can then alter those positions manually by using the **Position** function under each section. In this example we are going to add two new sections to capture the control surfaces and the change of sweep angle.

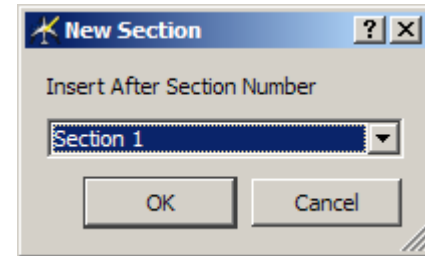


Figure 8 - New Section

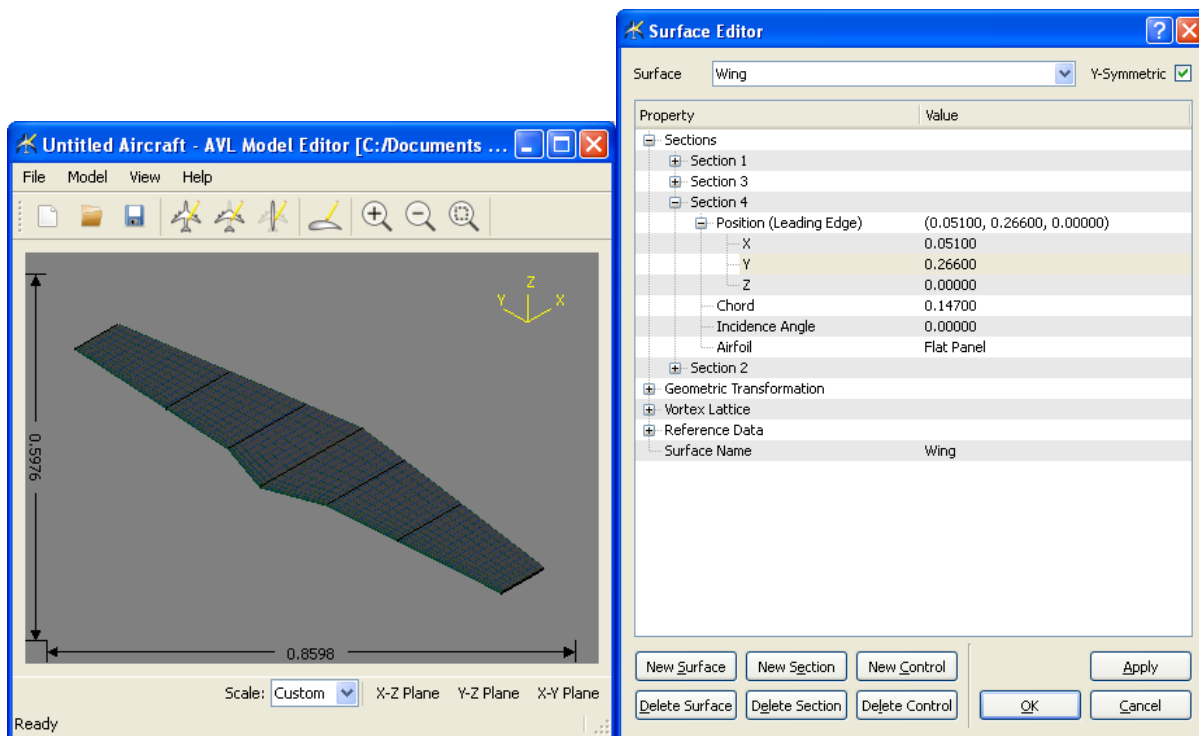


Figure 9 - Defining New Sections

4.3.2 Control Surfaces

Control surfaces are defined in between at least two sections. A control surface can span multiple sections. To add a control surface, use the **New Control** feature in the **Surface Editor**. When defining a control surface on a symmetric surface, the control surface will also be duplicated. Control surfaces are highlighted in yellow on the surface they are defined on.

In this example, we will add an aileron between Section 2 and Section 4.

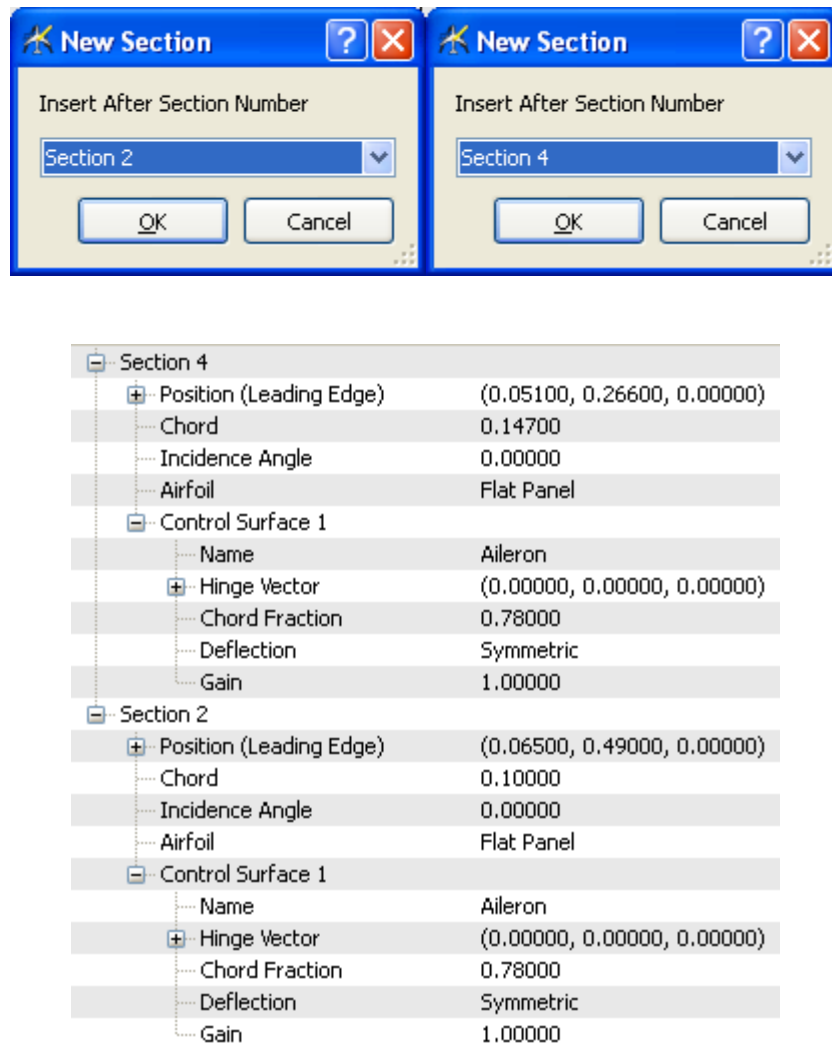


Figure 10 - Adding a Control Surface

The section that were selected now have a control surface selection. To make a control surface you must define an endpoint to the control surface. Simply create another control surface.

Defining a control surface consists of the following elements: **Name**, **Hinge Vector**, **Chord Fraction**, **Deflection**, and **Gain**.

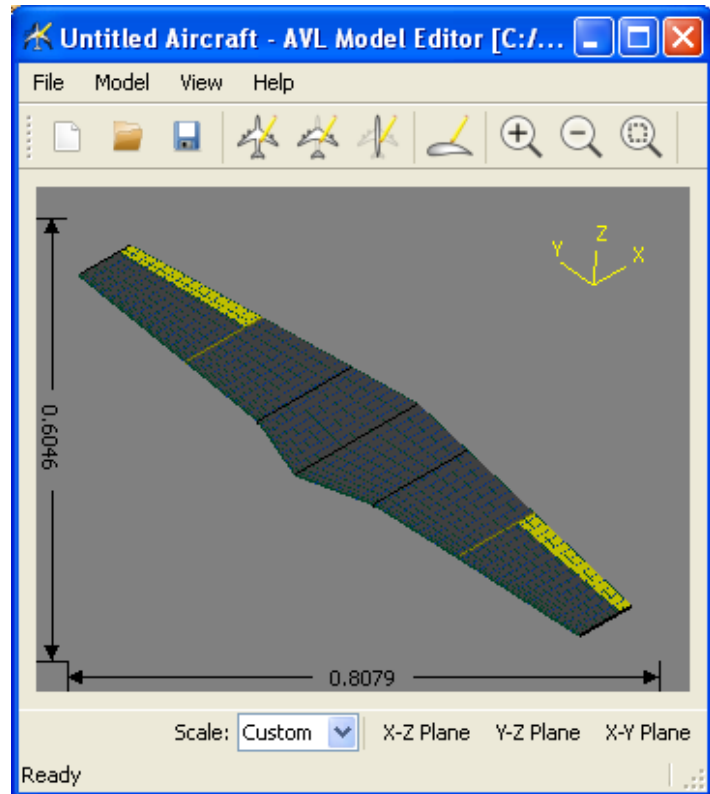


Figure 11 - Control Surface Selection

4.3.2.1 Hinge Vector

The Hinge Vector on the control surface is a non-functional piece of code in the AVL. It is designed to allow control surfaces to be rotated at a different location from the defined control surface leading edge.

4.3.2.2 Chord Fraction

Chord Fraction defines the size and location of the control surface. Positive values define the control surface extending from Chordfraction/Chord to the trailing edge. Negative values define the control surface extent from the LE to $-Chordfraction/Chord$.

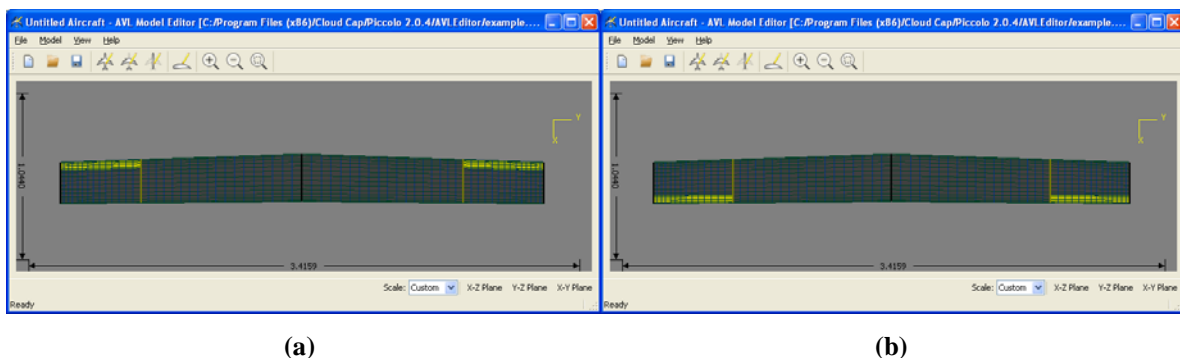


Figure 12 - (a) Chord fraction value of -0.2. Control surface extends from the LE to .2/C down the span. (b) Chord fraction value of 0.8. Control surface extends from 0.8/C to the TE of the surface.

4.3.2.3 Deflection

Used when defining control surfaces on a symmetric surface. This allows the controls surface rotations to be defined as symmetric (i.e. elevators) or asymmetric (i.e. ailerons).

4.3.2.4 Gain

The Control Surface gain is used to define direction of rotation of the control surface. The direction of rotation is defined by the right hand rule based off the direction of surface definition. Piccolo defines positive control surface motion as one that produces lift, i.e. aileron down is positive rotation/rudder right, creating positive yaw motion.

Surfaces defined from left to right, result in a positive control surface rotation following the right hand rule. Gain values should be set to 1 for this case. Surfaces defined from right to left result in negative control surface rotation. The gain value should be set to -1.

4.3.3 Geometric Translation

When creating new surfaces it is easiest to define the leading edge of the surface at the 0,0,0 section position, and use the geometric transformation tool to position the surface in it proper location. This allows you to easily modify the position of the surface without having to recalculate and reenter all the surfaces section positions.

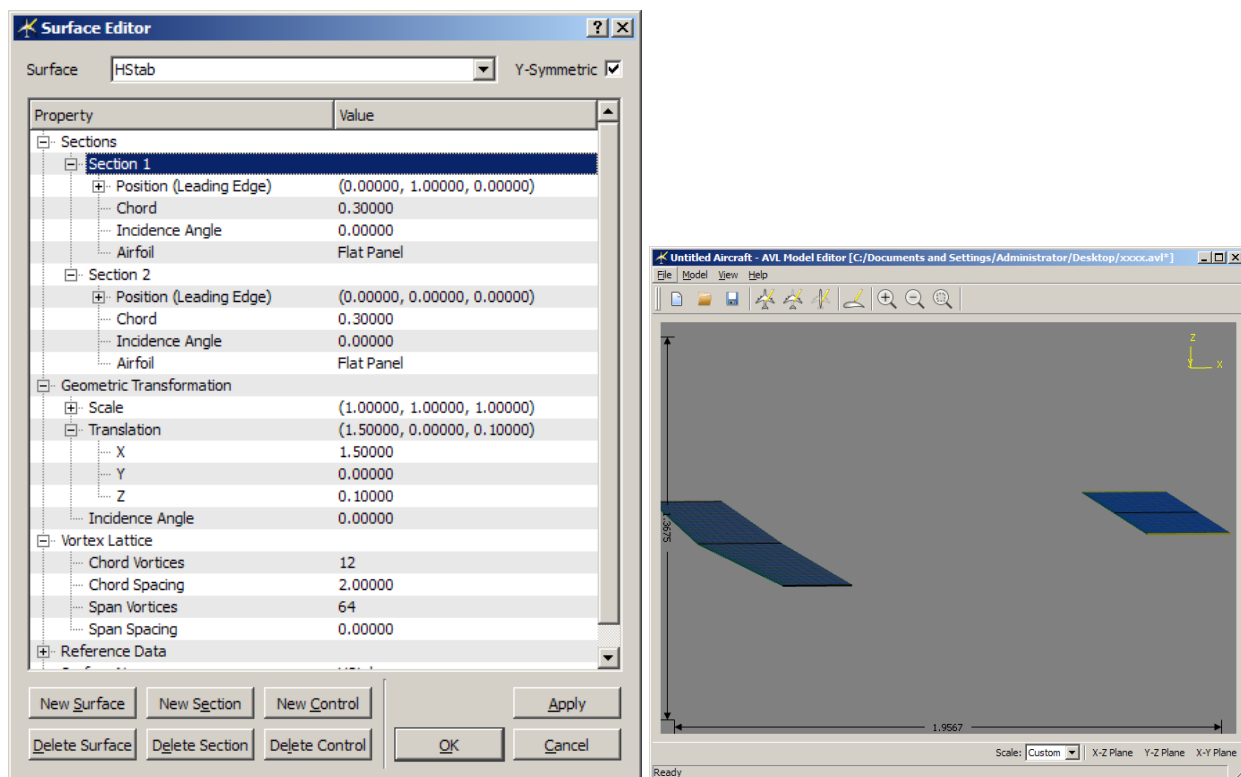


Figure 13 - Geometric Translation

4.3.4 Vortex Lattice

The vortex lattice feature defines how many vortices are used to define the wing. There are four parameters that define how the vortices are defined on the surface.

- **Chord/Span Vortices:** Number of vortices in the chordwise/spanwise direction.
- **Chord/Span Spacing:** A number from -3 to 3 defining either a linear, sine, or cosine spacing of the vortices. An intermediate parameter will result in a blended distribution between the two parameters, i.e. a value of 2.5 will result in a blend between sine and equal spacing.

Table 1 -Chord/Span Spacing

Parameter	Spacing	Spacing Representation
3.0	equal	
2.0	sine	
1.0	cosine	
0	equal	
-1.0	cosine	
-2.0	- sine	
-3.0	equal	

For optimum results, the following rules defined in the AVL documentation should be followed:

- In a standard VL method, a trailing vortex leg must not pass close to a downstream control point; else the solution will be garbage. In practice, this means that surfaces which are lined up along the x direction (i.e. have the same or nearly the same y,z coordinates), must have the same spanwise vortex spacing. AVL relaxes this requirement by employing a finite core size for each vortex on a surface which is influencing a control point in another surface (unless the two surfaces share the same INDEX declaration). This feature can be disabled by setting the core size to zero in the OPER sub-menu, Option sub-sub-menu, command C. This reverts AVL to the standard VL method.
- Spanwise vortex spacings should be "smooth", with no sudden changes in spanwise strip width. Adjust Nspan and Sspace parameters to get a smooth distribution. Spacing should be bunched at dihedral and chord breaks, control surface ends, and especially at wing tips. If a single spanwise spacing distribution is specified for a surface with multiple sections, the spanwise distribution will be fudged as needed to ensure that a point falls exactly on the section location. Increase the number of spanwise points if the spanwise spacing looks ragged because of this fudging.

- If a surface has a control surface on it, an adequate number of chordwise vortices N_{chord} should be used to resolve the discontinuity in the camberline angle at the hingeline. It is possible to define the control surface as a separate surface entity. The optional X1, X2 parameters of the airfoil and aFile keywords are specifically intended for this purpose. Cosine chordwise spacings then produce bunched points exactly at the hinge line, giving the best accuracy. The two surfaces must be given the same index and the same spanwise point spacing for this to work properly. Such extreme measures are rarely necessary in practice, however. Using a single surface with extra chordwise spacing is usually sufficient.
- When attempting to increase accuracy by using more vortices, it is in general necessary to refine the vortex spacings in both the spanwise and in the chordwise direction. Refining only along one direction may not converge to the correct result, especially locally wherever the bound vortex line makes a sudden bend, such as a dihedral break, or at the center of a swept wing. In some special configurations, such as an unswept planar wing, the chordwise spacing may not need to be refined at all to get good accuracy, but for most cases the chordwise spacing will be significant.

4.4 Body Editor

AVL has the capability of modeling bodies, i.e. fuselages and nacelles, via source and doublet filaments according to the slender body theory. AVL does not recommend modeling bodies if they are expected to have little influence on the aerodynamics of the system. AVLEditor provides a tool for quickly generating bodies. Bodies are saved as a separate text file. This file needs to be located in the same directory as the AVL files.

To create a new body, select **New Body...** from the pull down menu. Bodies are defined by using the **x** position, **z** position, and **Area**. As with the surfaces, the body can be scaled and translated. The vortices spacing and distribution is also defined similarly to the surfaces.

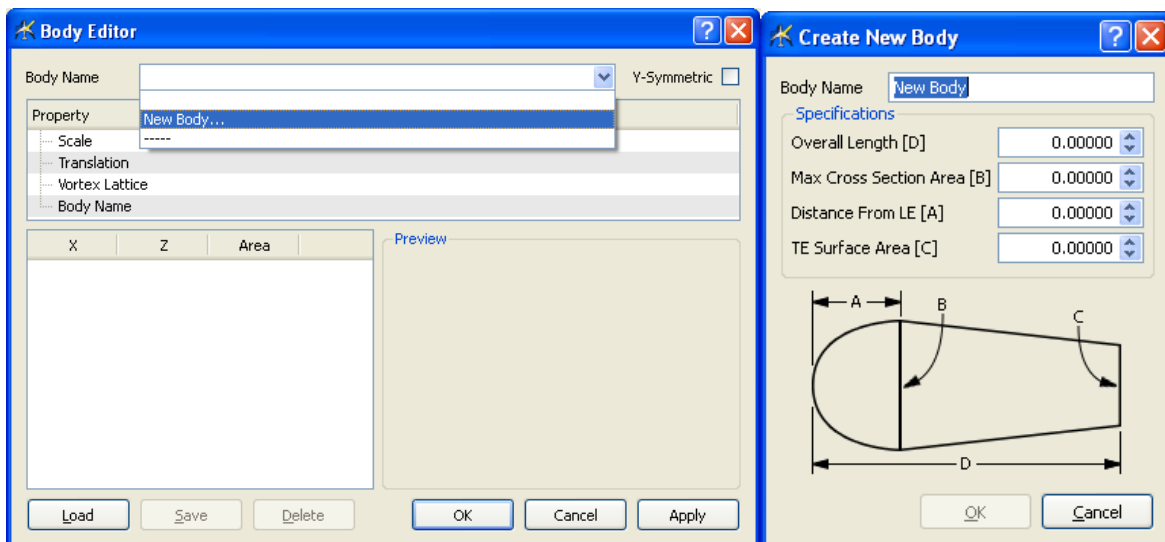


Figure 14 - Body Editor

4.5 XFOIL Analysis

XFOIL, also by Mark Drela et al., allows you to calculate airfoil properties. It has a sophisticated model of low speed aerodynamics (low Reynolds number data can be generated). XFOIL combines high-order panel methods with a fully-coupled viscous/inviscid interaction method.

XFOIL like AVL has its limitations, look out for lack of convergence (possibly Re too low or airfoil data not smooth enough).

AVLEditor has created an interface to XFOIL to help analyze the CDCL distributions for the airfoils. For more information regarding the specifics of XFOIL please reference the [XFOIL documentation](#).

For each surface, AVLEditor uses XFOIL to calculate the CDCL curves and Cla curve based off the surface Reynolds number. When running XFOIL Analysis you have the option for specifying Minimum Reynolds Number and Maximum Iterations. Specifying a minimum Reynolds number helps XFOIL converge when actual Reynolds numbers are low. The Maximum iteration prevents XFOIL from trying to solve a non-converging solution.

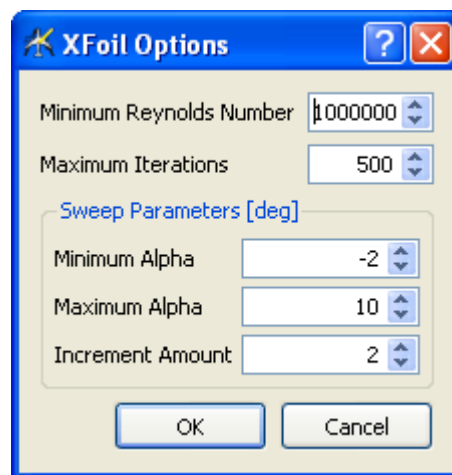


Figure 15 - XFOIL Options

The menu in **Figure 16** also has the option of manually specifying the CDCL and CLa curves for each of the surfaces. This allows you to use empirical data to specify the curves.

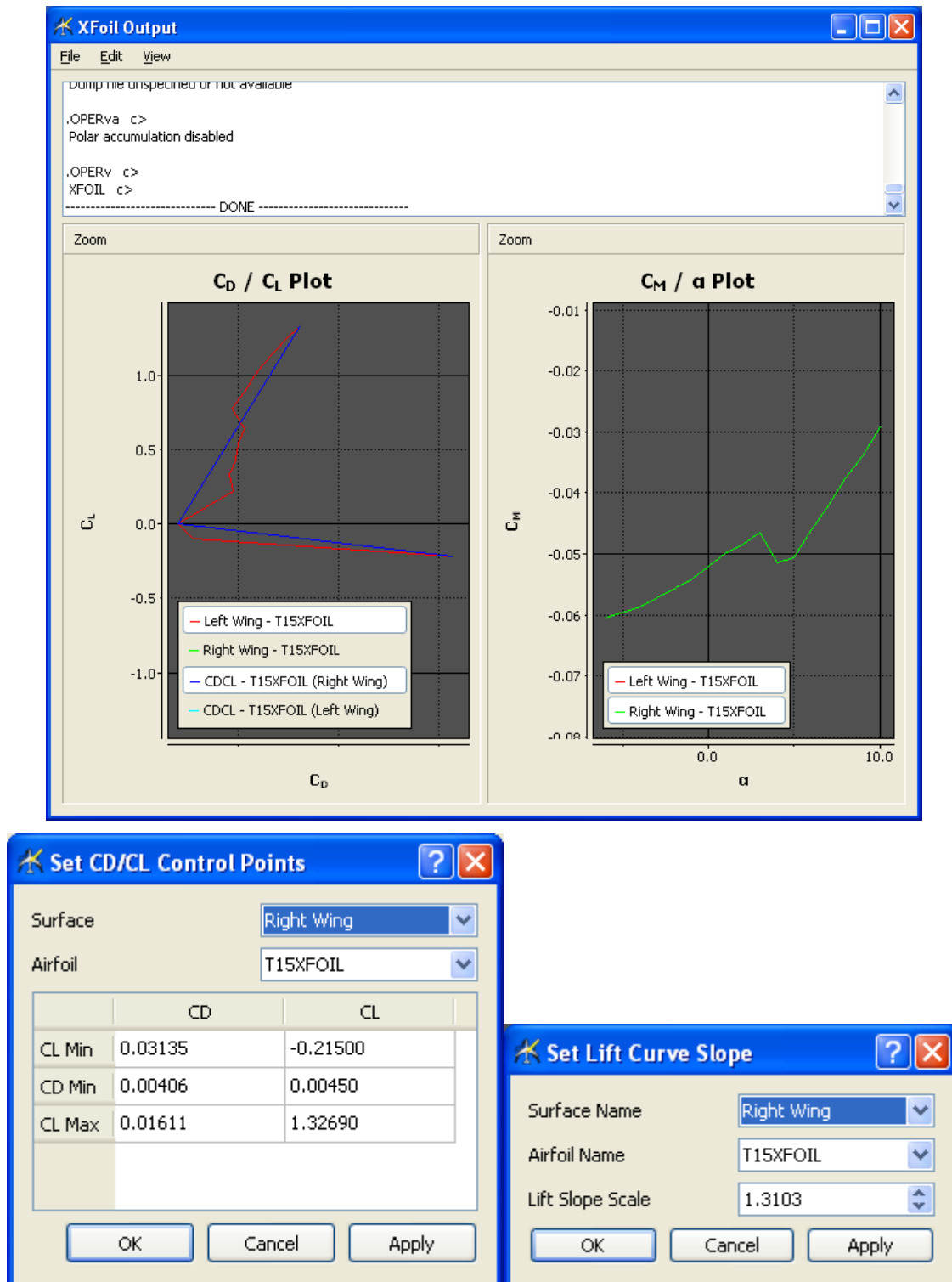


Figure 16 - Specifying Curves

4.6 AVL Analysis

The final step in aero-modeling process is to run the AVL code. From **AVL Model Editor** window menu, go to **Model » AVL Analysis**. This brings up the **AVL Output** window. From here you can run AVL analysis on your aircraft and review the results in the plotting functions.

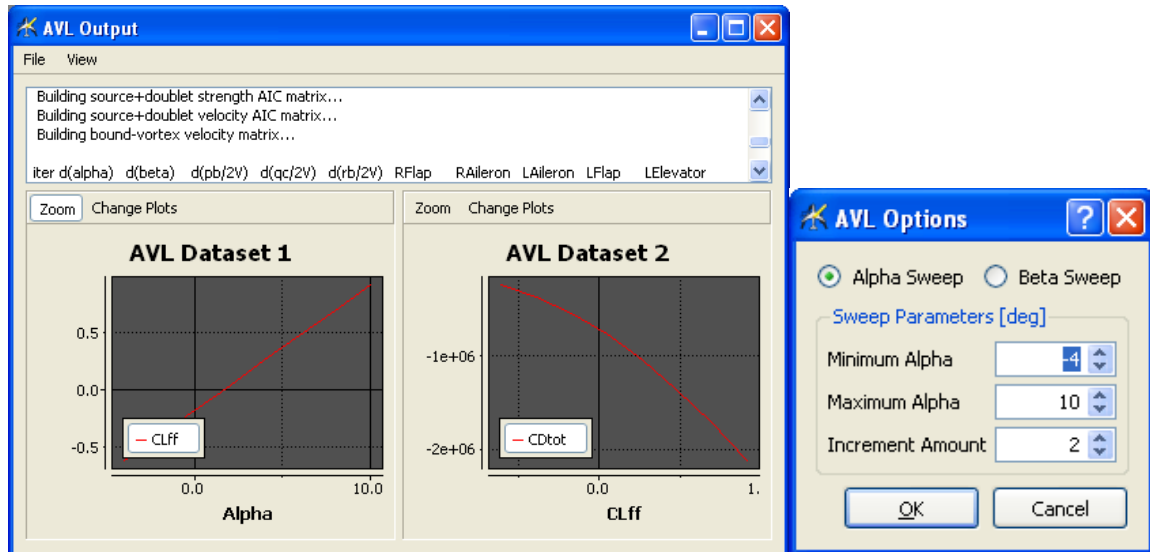


Figure 17 - AVL Analysis

There are 3 basic options for running an AVL analysis that define the alpha sweep:

- Minimum Alpha
- Maximum Alpha
- Increment amount

***Note:** Currently the Beta Sweep function is non-functional and should not be used.*

4.7 Control Surfaces

Specify an association between the control surfaces in PCC and the AVL control surfaces:

// Left Aileron	AVLEditor XML output
Channel_d1=0	<surface_1> Laileron </surface_1>
// Right Aileron	<surface_2> Raileron </surface_2>
Channel_d2=5	<surface_3> Rudder </surface_3>
// Rudder	<surface_4> Elevator </surface_4>
Channel_d3=3	
// Elevator	
Channel_d4=1	

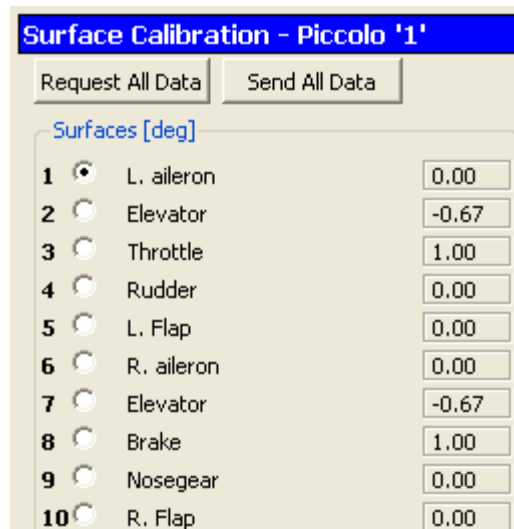


Figure 18 - Surface Calibration Window in PCC

Channel_dn is the AVL control surface. For assignments, look in the xml file. They are listed in numerical order.

If using control surface mixing such as ruddervators (or elevons) you can specify the left and right ruddervator channels. For example, aerosonde.avl implements separate ailerons and ruddervators, and the Aerosonde.txt file assigns the AVL control surfaces d1..d4 to the appropriate Piccolo channels:

- // Right aileron
- Channel_d1=5
- // Left aileron
- Channel_d2=0
- //Right ruddervator
- Channel_d3=8
- //Left ruddervator
- Channel_d4=3

5 Inertia Model

The Piccolo Simulator can take in inertia data directly with the following parameters:

- Roll_Inertia=
- Pitch_Inertia=
- Yaw_Inertia=
- Roll_Yaw_Coupled_Inertia=

The inertia model includes the mass and rotational inertia of the whole airplane. The mass depends on the initial mass and the amount of fuel that has been burned. Rotational inertia is based on the inertias of the X, Y, and Z axis. Additionally, one cross component, X to Z is included.

Table 2 - Inertia Parameter Values

Value	Meaning	Default
Gross_Mass	Mass of the entire aircraft, including fuel, in kg.	Required
Empty_Mass	Mass of the aircraft with no fuel, kg	Required
Roll_Inertia	Inertia about the X axis in kg-m ² .	Required
Pitch_Inertia	Inertia about the Y axis in kg-m ² .	Required
Yaw_Inertia	Inertia about the Z axis in kg-m ² .	Required
Roll_Yaw_Coupled_Inertia	XZ inertia coupling in kg-m ² .	0.0

In addition to mass and moments of inertia, the inertia model also includes the acceleration effects due to the avionics being mounted at a certain distance from the aircraft CG, as well as the frame transformations for inertial measurements due to misalignment between avionics and aircraft body frame. The parameters are listed in **Table 3**.

Table 3 - Avionics (or IMU) Mounting Geometry

Value	Meaning	Default
IMU_Sensor_Roll_Angle	The roll angle of the avionics with respect to the body frame, in deg.	0.0
IMU_Sensor_Pitch_Angle	The pitch angle of the avionics with respect to the body frame, in deg.	0.0
IMU_Sensor_Yaw_Angle	The yaw angle of the avionics with respect to the body frame, in deg.	0.0
IMU_Sensor_Position_X	The X-axis component of the position vector of the avionics' center with respect to the aircraft CG, in m.	0.0
IMU_Sensor_Position_Y	The Y-axis component of the position vector of the avionics' center with respect to the aircraft CG, in m.	0.0
IMU_Sensor_Position_Z	The Z-axis component of the position vector of the avionics' center with respect to the aircraft CG, in m.	0.0

An inertia by component spreadsheet is available in the [AVLCCT.zip](#) archive. It can handle V-tails and can be used as an empirical or exact model or it can calculate inertias with a basic point mass + block model with the parameters found in the *Appendix*. (Values are in kg and m.)

6 Propulsion Model

The propulsion model estimates the loads (three forces and three moments) generated by the propulsion system on the simulated aircraft as a function of the throttle setting, engine data, actuator data, and aircraft state. The Simulator supports left and right propulsion units. Each of the parameter names given in the following sections should be preceded by either a “Left_ “or a “Right_ “depending on which propulsion unit is intended.

Each of the left and right propulsion units consists of an engine and an actuator or combination of two actuators (**Figure 19**). The engine produces mechanical work for the actuator(s) while these generate forces and moments that are applied to the airframe.

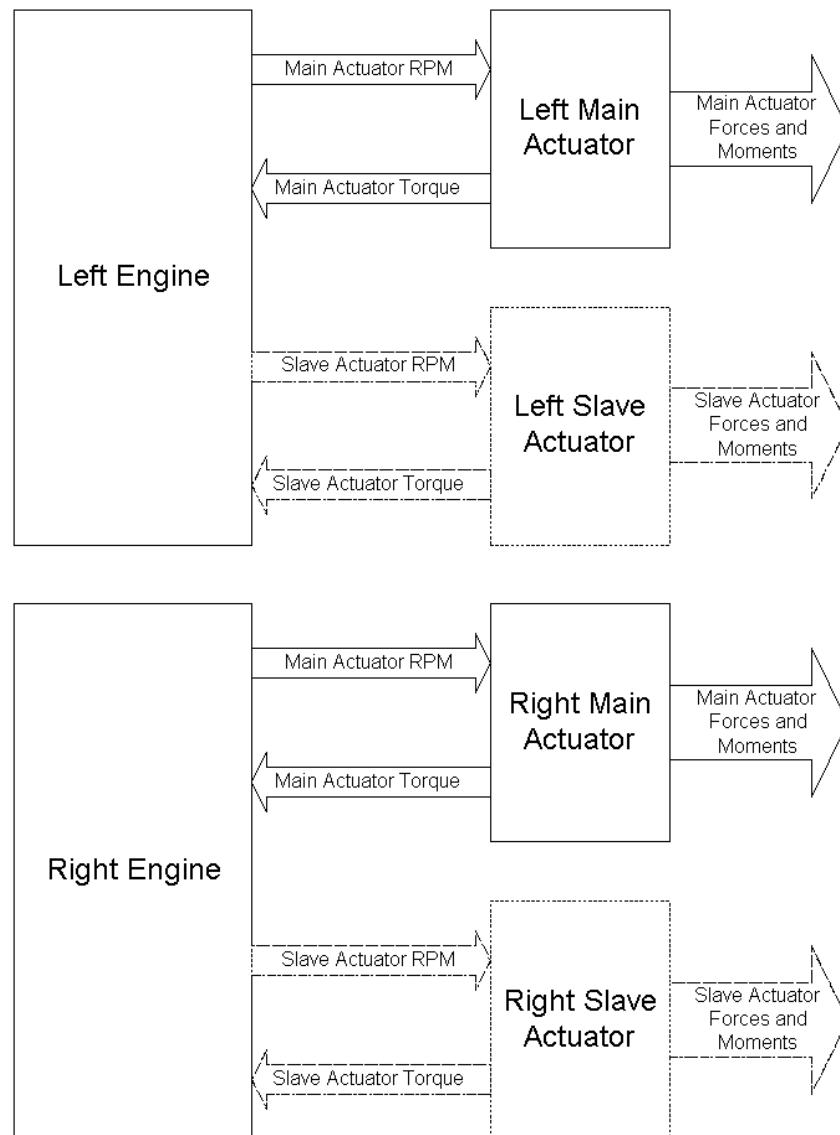


Figure 19 - Propulsion Model Structure

6.1 Engine Models

The engine is the component that transforms the internal energy (chemical or electrical) stored on-board the aircraft into mechanical work used to drive the actuator. The Simulator propulsion system is designed to support multiple types of engine models. The piston engine and the electric motor engine are the models currently available in the Simulator.

Before defining the parameters for a specific engine model, the general engine parameters shown in **Table 4** should be included in the aircraft configuration file. After specifying the engine type, the engine model will be completed with model-specific parameters.

Table 4 - General Engine Parameters

Value	Meaning	Default
Engine_Type	Index of the engine model to be used: 0 = Piston engine 1 = Electric motor	0 (piston engine)

6.1.1 Piston Engine Model

The piston engine model calculates the torque and power for a given engine based on the throttle, RPM, and ambient pressure. It is based upon a simple one-dimensional look up table which relates wide-open-throttle (WOT) power to RPM.

To calculate the power at each time step, the engine model uses the current RPM to calculate the WOT power. That power number is then multiplied by the ratio of the ambient pressure to standard sea level pressure, and also by the throttle position (which is a number from 0 to 1). Fuel flow is calculated using a single number, the brake-specific fuel consumption, which relates the fuel consumption to the amount of power produced.

The piston engine model suffices for purposes of simulation. More advanced models can be constructed, but they require more detailed knowledge of the actual engine implementation.

Estimating the engine power by a look up table is done by reading in data from the file indicated in the parameter list. This file must contain two columns of ASCII data. The first column is the RPM, and the second column is the corresponding WOT power, in Watts. The RPM must be given in ascending order. The following is a sample engine look-up table for a Honda Mini 4-stroke engine:

Honda Mini 4-stroke GX-31

Look-up Table

Wide open throttle

RPM Power[W]

```
0 0.00
1000 75.00
4000 730.00
5000 950.00
6000 1050.00
7000 1100.00
8000 1075.00
9000 950.00
10000 100.00
```

The piston engine is a rotating engine. As such, the rotational acceleration of the engine is calculated from the difference between the torque produced by the engine and the torque consumed by the actuator(s), and the rotational inertia of the engine shaft, gearbox, and actuator(s). The engine angular speed is obtain by time-integration of the angular acceleration.

In a rotating engine, the piston engine model incorporates an optional governor that is modeled as a proportional-integral feedback loop that regulates the engine angular speed using throttle.

Table 5 - Piston Engine Parameter Values

Value	Meaning	Default
Engine_Channel	Throttle servo channel assignment	2 for left engine, 7 for right engine
Engine_CutOffThrottle	The throttle fraction at which the engine stops running	0.0
Engine_Inertia	Moment of inertia of the engine rotor, in $\text{kg}\cdot\text{m}^2$	0.0
Engine_FrictionTorque	Torque consumed by internal friction, in $\text{N}\cdot\text{m}$	0.0
Engine_GovernorGainPro	RPM governor proportional gain. By default the governor is disabled. To enable it, set this parameter to non-zero value.	0.0 (governor disabled)
Engine_GovernorGainInt	RPM governor integral gain	0.0
Engine_GovernorRPM	Governor engine RPM command	0.0
Engine_GovernorRPMWindow	Governor activation window (RPM regulation starts when $\text{RPM} > \text{RPMcmd} - \text{RPMwindow}$)	0.0
Engine_LUT	File name for the Look up table of the engine WOT power (in Watts) versus RPM curve.	None, parameter is required
Engine_BSFC	Brake specific fuel consumption for the engine, in grams of fuel per hour per Kilowatt of power	0.0

6.1.2 Electric Motor Model

The electric motor model implements the DC motor equations that provide the motor torque as a function of input voltage and shaft RPM. The model does not employ look-up tables, instead it relies on the motor characteristics typically provided by the DC motor manufacturer. The motor input voltage is assumed to be linear with the throttle input. Zero throttle results in a zero-voltage input to the motor, and full throttle (=1) results in the nominal voltage input to the motor.

The electric motor is a rotating engine. It includes rotational acceleration based on produced torque, load torque, and inertia of the rotating masses. It also includes an optional RPM governor model.

Table 6 - Electric Motor Parameter Values

Value	Meaning	Default
Motor_Channel	Throttle servo channel assignment	2 for left engine, 7 for right engine
Motor_Inertia	Moment of inertia of the engine rotor, in $\text{kg}\cdot\text{m}^2$	0.0
Motor_GovernorGainPro	RPM governor proportional gain. By default the governor is disabled. To enable it, set this parameter to non-zero value.	0.0 (governor disabled)
Motor_GovernorGainInt	RPM governor integral gain	0.0

Motor_GovernorRPM	Governor engine RPM command	0.0
Motor_GovernorRPMWindow	Governor activation window (RPM regulation starts when $RPM > RPM_{cmd} - RPM_{window}$)	0.0
Motor_NominalInputVoltage	The nominal voltage the motor batteries will provide, in Volts (V).	0.0
Motor_TorqueConstant	Typically represented by K_m or K_i , it is the motor torque per coil current, in Newton-meters per Ampere ($N \cdot m/A$).	Required, if RPMConstant not defined
Motor_RPMConstant	Typically represented as K_v , it is the no-load shaft speed per input voltage, in rotations per minute per Volt (rpm/V).	Required, if TorqueConstant not defined
Motor_NoLoadCurrent	Also known as idle current (I_0) it is the coil current when running with no shaft load, in Amperes (A).	Required
Motor_TerminalResistance	Typically represented as R_i , it is the resistance of the armature, given in Ohms (Ω).	Required
Motor_ThermalResistance	The characteristic value of the thermal transition resistance from winding to housing and from housing to ambient, in degrees Kelvin per Watt (K/W).	0.0

If the aircraft model is using an electric motor, fuel consumption displayed in the Simulator dialog is represented by the coil current shown in amperes.

The following is an example of the configuration parameters for a Kohler Actro 32-4 electric motor:

```
// Electric motor Kohler Actro 32-4
Left_Engine_Type=1
Left_Motor_Channel=4
Left_Motor_Inertia=0.03
Left_Motor_NominalInputVoltage=42.0
Left_Motor_RPMConstant=415
Left_Motor_NoLoadCurrent=1.21
Left_Motor_TerminalResistance=0.055
Left_Motor_ThermalResistance=1.2
```

6.2 Engine Actuator Models

By engine actuator we are referring to the engine subsystem that creates loads (three forces and three moments) on the airframe. Each of the left and right engines will have a corresponding left or right actuator, respectively. Each engine may have a single actuator (propeller or helicopter rotor) or a predefined combination of main actuator and slave actuator (helicopter main rotor and helicopter tail rotor). In the case of a combo actuator, both the main and the slave actuator will rotate at fixed distinct ratios of the engine angular speed.

The RPM displayed in the Simulator dialog is from the actuator, not the engine. In the case of a combo actuator, the RPM displayed is from the main actuator (the main rotor on a helicopter combo, for example).

Before defining the parameters for a specific actuator model, the parameters in **Table 7** should be included in the aircraft configuration file.

Table 7 - General Actuator Parameters

Value	Meaning	Default
Actuator_Type	Index of the actuator model to be used: 0 = Fixed-pitch propeller 1 = Simple rigid rotor 2 = Helicopter main rotor 3 = Helicopter tail rotor 4 = Helicopter combo main+tail (geared to the same engine)	0 (propeller)

After specifying the actuator type, the actuator model will be completed with model-specific parameters.

6.2.1 Fixed-Pitch Propeller Model

The propeller model calculates the thrust and torque of the propeller based on the RPM, and forward speed of the aircraft. It uses a propeller look up table that gives the coefficient of thrust and coefficient of power as a function of the advance ratio.

The Simulator calculates the advance ratio, and uses the look up table to determine C_t and C_p . C_t is used to calculate the thrust from the propeller. C_p is used to calculate the power absorbed by the propeller.

Estimating the propeller performance by look up table is done by reading in data from the file indicated in the parameter list. This file must contain three columns of ASCII data. The first column is the advance ratio, and the second column is the coefficient of power, and the third column is the coefficient of thrust. The advance ratio must be given in ascending order. A sample look-up table for an APC 20x8 propeller is shown in the *Appendix*.

The torque consumed by the propeller is applied to the propulsion system rotational dynamics. The thrust produced by the propeller is transformed to forces and moments in aircraft body axes, at aircraft CG location. A propeller definition includes the parameters in the aircraft configuration file shown in **Table 8**.

Table 8 - Propeller Parameter Values

Value	Meaning	Default
Prop_X	The x-axis location of the actuator. Positive if actuator is ahead of aircraft CG, in meters (m).	0.0
Prop_Y	The y-axis location of the actuator. Positive if actuator is to the right of aircraft CG, in meters (m).	0.0
Prop_Z	The z-axis location of the actuator. Positive if actuator is below the aircraft CG, in meters (m).	0.0
Prop_Tilt	Aircraft-to-actuator frame transformation pitch angle, in deg. Positive if actuator is pitched up from aircraft axes, or negative if actuator is pitched down.	0.0
Prop_Pan	Aircraft-to-actuator frame transformation yaw angle, in deg. Positive if actuator is yawed to the right of the aircraft axes, or negative if the actuator is yawed to the left.	0.0

Prop_Diameter	Propeller diameter, in m.	Required, if Prop_Radius not defined
Prop_Radius	Propeller radius, in m.	Required, if Prop_Diameter not defined
Prop_Inertia	Propeller moment of inertia, in $\text{kg}\cdot\text{m}^2$.	0.001
Prop_GearRatio	Engine-to-propeller gear ratio. Ratio is greater than 1 if prop rotates slower than the engine (gear reduction).	1.0
Prop_Sense	Propeller sense of rotation (set to 1 for clockwise X-axis rotation, or to -1 for counter-clockwise X-axis rotation). Propeller always rotates about its X-axis.	1
Prop_LUT	File name for the Look up table of the propeller performance.	Required

Several parameters that define the actuator position and orientation are graphically represented in

Figure 20.

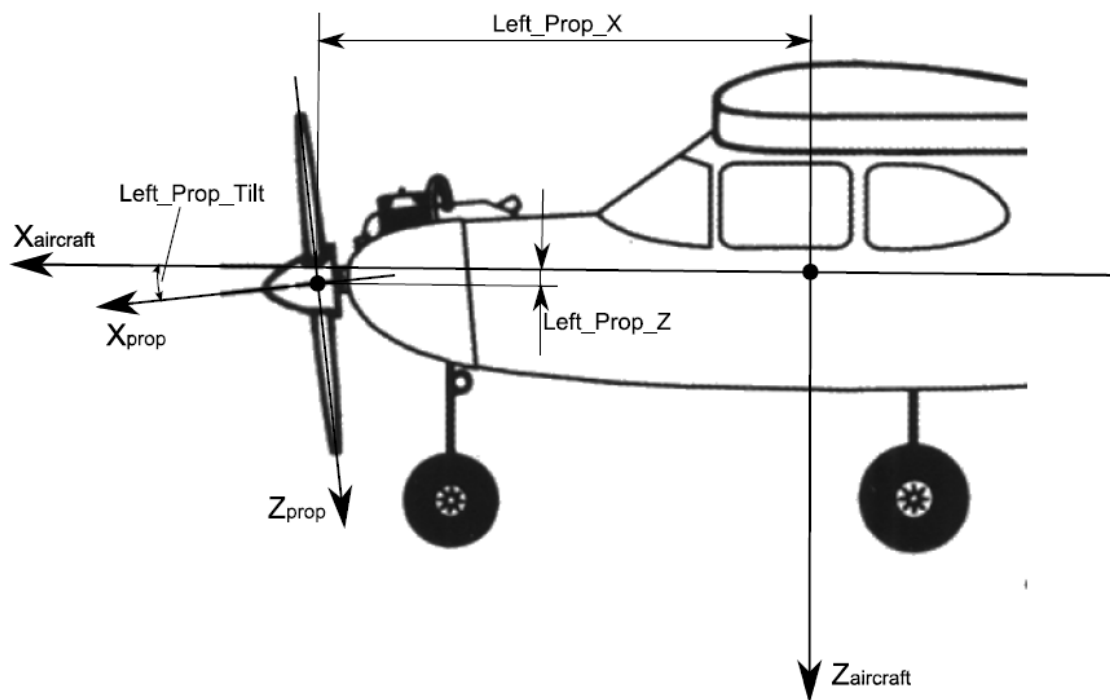


Figure 20 - Propeller Actuator Position and Orientation

The following tools are available to help with propeller modeling:

- [CreatingPropModels.pdf](#)
- [PropModelling.zip](#)

The *Creating Prop Models* document contains a method for approximating prop inertia as well as a small database of estimated prop inertias. A reasonable prop inertia value has proven important for determining altitude controller gains.

6.2.2 Simple Rigid Rotor

The simple rigid rotor model uses momentum theory to calculate the induced velocity and thrust of a variable-pitch rotor. The rotor blades are assumed to have no twist and constant chord. The induced velocity calculation accounts for ground effect.

The rigid rotor model user-configurable parameters are shown in **Table 9**.

Table 9 - Rigid Rotor Parameter Values

Value	Meaning	Default
Rotor_X	The x-axis location of the actuator. Positive if actuator is ahead of aircraft CG, in meters (m).	0.0
Rotor_Y	The y-axis location of the actuator. Positive if actuator is to the right of aircraft CG, in meters (m).	0.0
Rotor_Z	The z-axis location of the actuator. Positive if actuator is below the aircraft CG, in meters (m).	0.0
Rotor_Tilt	Aircraft-to-actuator frame transformation pitch angle, in deg.	0.0
Rotor_Pan	Aircraft-to-actuator frame transformation yaw angle, in deg.	0.0
Rotor_Diameter	Rotor diameter, in m	Required, if Rotor_Radius not defined
Rotor_Radius	Rotor radius, in m	Required, if Rotor_Diameter not defined
Rotor_Inertia	Rotor moment of inertia, in kg*m ²	0.001
Rotor_GearRatio	Engine-to-rotor gear ratio	1.0
Rotor_Sense	Rotor sense of rotation (set to 1 for clockwise X-axis rotation, or to -1 for counter-clockwise X-axis rotation). Rotor always rotates about its X-axis.	1
Rotor_NumberOfBlades	Number of blades	0
Rotor_BladeLiftSlope	Slope of the coefficient of lift for the blade airfoil, per rad.	2 π
Rotor_BladeProfileDrag	Profile drag coefficient of the blade airfoil	0.0
Rotor_BladeChord	Blade chord, in m	0.0
Rotor_GroundEffect	Factor that defines the magnitude of ground effect, KGE, empirical	0.05
Rotor_IsVariableCollective	1 if rotor has variable collective controlled by a servo, or 0 if the rotor collective pitch angle is fixed	0
Rotor_CollectivePitch	If rotor collective is fixed, this defines the blade pitch angle, in deg.	0.0
Rotor_CollectiveChannel	If rotor has variable collective, this defines the collective input servo channel	3 for left rotor, 8 for right rotor

6.2.3 Helicopter Main Rotor

The helicopter main rotor model is derived from the simple rigid rotor model, but adds the following

- First-order flapping dynamics
- Secondary rotor representing the flybar with its own flapping equations
- 2-axis variable cyclic
- an assumed vertical orientation with respect to the aircraft

The main rotor model user-configurable parameters are shown in **Table 10**.

Table 10 - Helicopter Main Rotor Parameter Values

Value	Meaning	Default
MainRotor_X	The x-axis location of the actuator. Positive if actuator is ahead of aircraft CG, in meters (m).	0.0
MainRotor_Y	The y-axis location of the actuator. Positive if actuator is to the right of aircraft CG, in meters (m).	0.0
MainRotor_Z	The z-axis location of the actuator. Positive if actuator is below the aircraft CG, in meters (m).	0.0
MainRotor_Diameter	Rotor diameter, in m	Required, if MainRotor_Radius not defined
MainRotor_Radius	Rotor radius, in m	Required, if MainRotor_Diameter not defined
MainRotor_GearRatio	Engine-to-rotor gear ratio	1.0
MainRotor_Sense	Rotor sense of rotation (set to 1 for positive X-axis rotation, or to -1 for negative X-axis rotation). This controls the sense of the main rotor torque.	1
MainRotor_NumberOfBlades	Number of blades	0
MainRotor_BladeLiftSlope	Slope of the coefficient of lift for the blade airfoil, per rad.	2π
MainRotor_BladeProfileDrag	Profile drag coefficient of the blade airfoil	0.0
MainRotor_BladeChord	Blade chord, in m	0.0
MainRotor_GroundEffect	Factor that defines the magnitude of ground effect, KGE, empirical	0.05
MainRotor_IsVariableCollective	1 if rotor has variable collective controlled by a servo, or 0 if the rotor collective pitch angle is fixed	0
MainRotor_CollectivePitch	If rotor collective is fixed, this defines the blade pitch angle, in deg.	0.0
MainRotor_CollectiveChannel	If rotor has variable collective, this defines the collective input servo channel	3 for left rotor, 8 for right rotor
MainRotor_BladeHingeOffset	Blade flapping hinge offset from rotor hub, in m	0.0
MainRotor_BladeMass	Blade mass, in kg	0.0
MainRotor_BladeTwist	Blade tip twist angle, in deg.	0.0

MainRotor_TeeterDamping	Rotor teeter damping coefficient	0.0
MainRotor_MaxTPPTilt	Maximum tilt angle of the rotor tip path plane, in deg. applies to both main rotor and flybar flapping angles.	30 deg.
MainRotor_MaxAdvanceRatio	Maximum rotor advance ratio	0.2
MainRotor_FlybarRadius	Flybar radius in m.	0.0 (no flybar)
MainRotor_FlybarLiftSlope	Flybar blade airfoil lift curve slope, per rad.	2π
MainRotor_FlybarBladeChord	Flybar blade chord, in m	0.0
MainRotor_FlybarBladeSpan	Flybar blade span (outer radius – inner radius), in m.	0.0
MainRotor_FlybarBladeInertia	Flybar blade flapping inertia about its hinge, in $\text{kg}\cdot\text{m}^2$	0.0
MainRotor_FlybarCyclicGain	Cyclic pitch of flybar per cyclic pitch of main rotor blade.	0.0
MainRotor_CyclicFlybarTPPGain	Cyclic pitch of main rotor blade per flybar tip path plane tilt.	0.0
MainRotor_LatCyclicChannel	Servo channel for direct control of main rotor lateral cyclic.	0 left, 5 right
MainRotor_LonCyclicChannel	Servo channel for direct control of main rotor longitudinal cyclic.	1 left, 6 right
MainRotor_SwashplateType	Swashplate mixing type: 0 = no mixing 1 = SWH2 mixing 2 = SWH2-reversed mixing 3 = SWH4 mixing 4 = SR3 mixing 5 = SR3-reversed mixing 6 = SN3 mixing 7 = SN3-reversed mixing	0 (no mixing)
MainRotor_Servo1Channel	First mixed servo channel	0 left, 5 right
MainRotor_Servo2Channel	Second mixed servo channel	1 left, 6 right
MainRotor_Servo3Channel	Third mixed servo channel	4 left, 9 right
MainRotor_Servo4Channel	Fourth mixed servo channel	3 left, 8 right
MainRotor_CyclicSwashplateGain	Deflection of main rotor blade per deflection of swashplate.	0.0
MainRotor_BladeGripArm	Blade grip arm length, length units consistent with linear servo pushrod deflection calibrated in the avionics (cm typical).	0.0
MainRotor_SwashplateRadius	Swashplate radius at servo pushrod connection, length units consistent with linear servo pushrod deflection calibrated in the avionics (cm typical).	0.0

Several of the main rotor and flybar geometry parameters can be represented graphically on a simple top-view drawing of the main rotor shown in **Figure 21**.

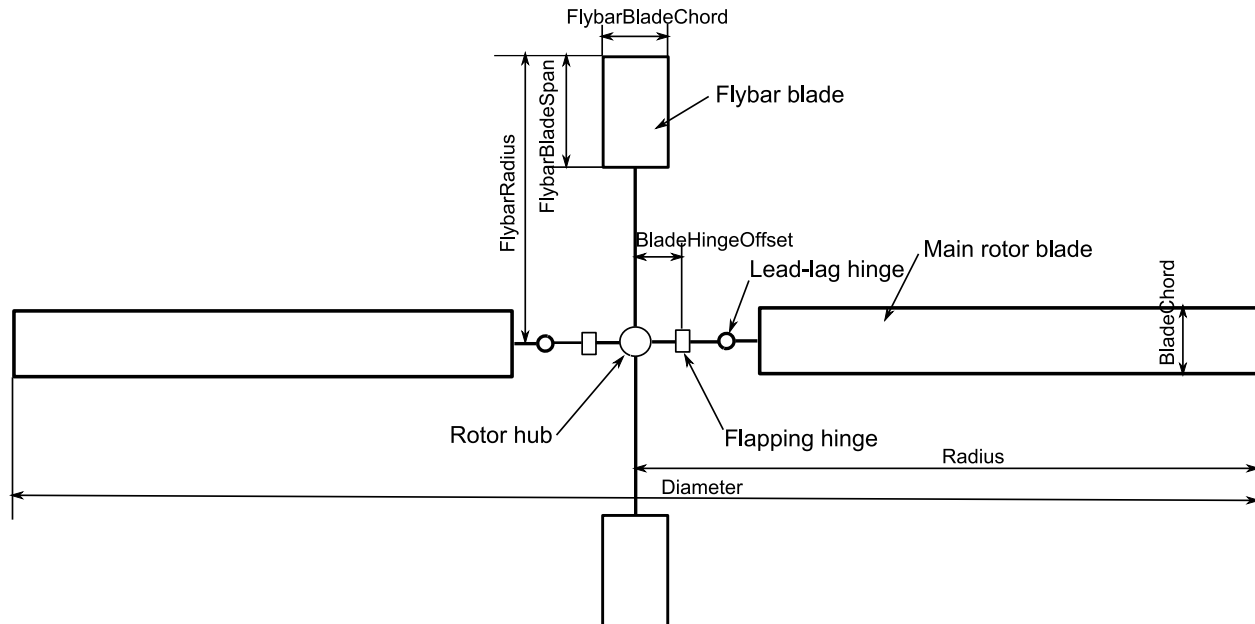


Figure 21 - Main Rotor and Flybar Geometry, Top View

A side view of the typical Bell-Hiller mixer is shown in **Figure 22**. The following dimensional elements are needed for the Simulator model:

- a and b = the mixer arm lengths from blade arm to flybar pushrod and to swashplate pushrod respectively;
- m = the main blade grip arm, see *MainRotor_BladeGripArm* parameter;
- n = the flybar blade grip arm or offset of the swashplate pushrod link to Hiller bridge from main rotor shaft center axis.
- p = radius of the Hiller bridge or offset of the mixer arm pushrod link from main rotor shaft center axis.
- $r1$ = the offset of mixer arm pushrod link from main rotor shaft center axis.
- $r2$ = the offset of Hiller bridge pushrod link from main rotor shaft center axis.
- R = the outer swashplate radius, or offset of servo pushrod from main rotor shaft center axis, see *MainRotor_SwashplateRadius* parameter.

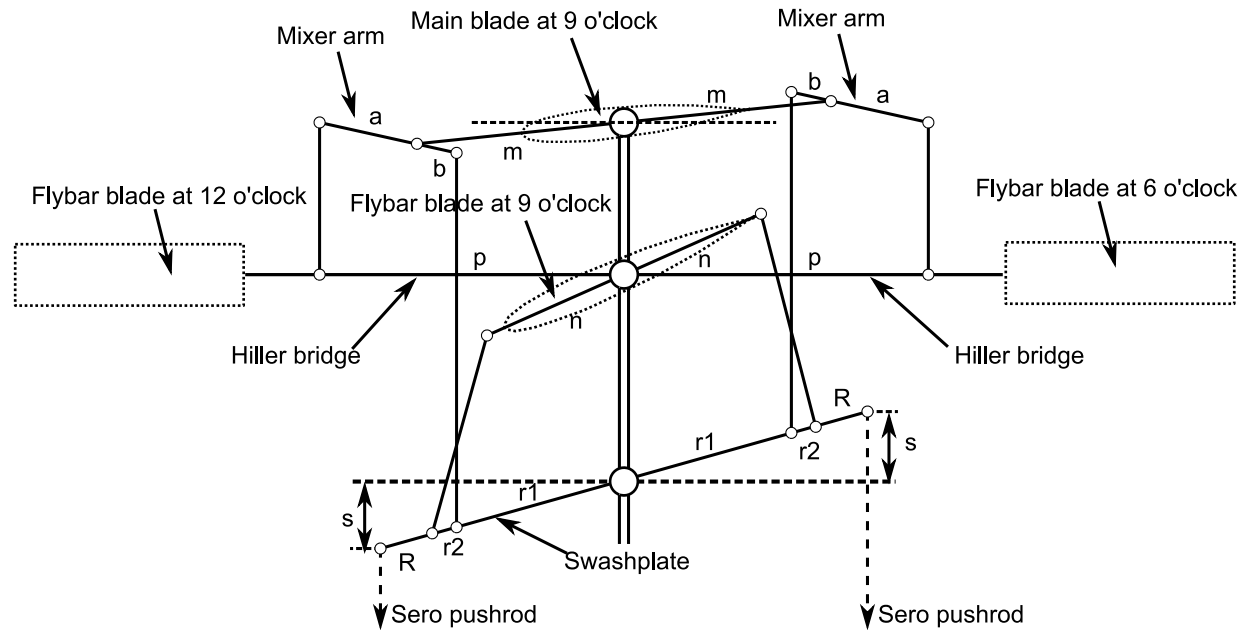


Figure 22 - Bell-Hiller Mixer Diagram

The aircraft model parameter `MainRotor_BladeGripArm` is element m shown in **Figure 22**. The parameter `MainRotor_SwashplateRadius` is element R . The parameter `MainRotor_FlybarCyclicGain` represents the change in flybar cyclic pitch angle per unit change of the main rotor cyclic pitch angle. It can be calculated from the dimensional elements in the diagram using the following formula:

$$\text{FlybarCyclicGain} = \frac{a+b}{a} \cdot \frac{m}{n} \cdot \frac{r_2}{r_1}$$

The parameter `MainRotor_CyclicFlybarTPPGain` represents the change in main rotor blade cyclic angle per unit change of the flybar tip path plane angle. It can be calculated from the parameters in the diagram using the following formula:

$$\text{CyclicFlybarTPPGain} = \frac{b}{a+b} \cdot \frac{p}{m}$$

The parameter `MainRotor_CyclicSwashplateGain` which represents the change in main rotor blade cyclic angle per unit change of swashplate tilt angle, can be calculated from the parameters in the diagram using the following formula:

$$\text{CyclicSwashplateGain} = \frac{a}{a+b} \cdot \frac{r_1}{m}$$

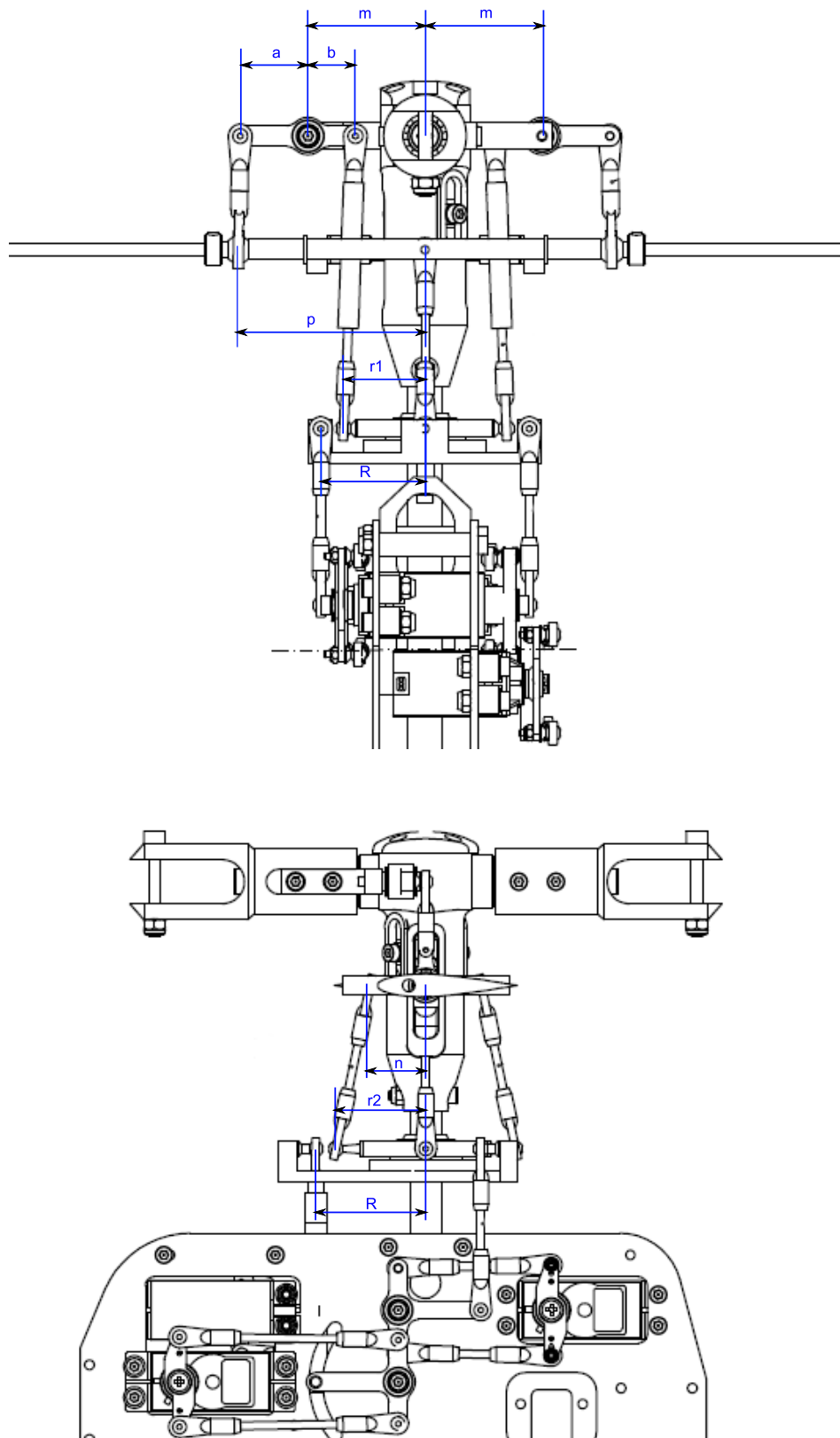


Figure 23 - Maxi-Joker 2 Rotor Head Example

The following is an example of the geometry parameters for the Maxi-Joker 2 rotor head shown in **Figure 23**.

Rotor head parameter	Value (on MaxiJoker 2), in mm
a	20.00
b	14.00
m	34.15
n	18.00
p	54.15
r ₁	24.50
r ₂	29.00
R	35.22

Using the formulas presented earlier, listed below are the following gains for the Maxi-Joker 2:

$$\text{FlybarCyclicGain} = 3.8177$$

$$\text{CyclicFlybarTPPGain} = 0.6529$$

$$\text{CyclicSwashplateGain} = 0.422$$

The typical R/C helicopter rotor – see Maxi-Joker 2 example in **Figure 23** – is not hingeless but not fully-articulated. The flapping hinge consists of an elastomer fitting that allows a limited amount of angular displacement of the rod connecting the two blade roots, acting both as a spring and as a damper. Because the rotor model implemented in the Simulator assumes an articulated rotor, some tuning of the `MainRotor_BladeHingeOffset` parameter would be required in order to improve the fidelity of the rotor flapping dynamics.

6.2.4 Helicopter Tail Rotor

The helicopter tail rotor model is derived from the simple rigid rotor, but it adds a proportional yaw damper, it removes the ground effect, and it assumes an orientation orthogonal to the aircraft's X-axis (90 deg. pan angle).

The tail rotor model has the following user-configurable parameters:

Table 11 - Helicopter Tail Rotor Parameter Values

Value	Meaning	Default
TailRotor_X	The x-axis location of the actuator. Positive if actuator is ahead of aircraft CG, in meters (m).	0.0
TailRotor_Y	The y-axis location of the actuator. Positive if actuator is to the right of aircraft CG, in meters (m).	0.0
TailRotor_Z	The z-axis location of the actuator. Positive if actuator is below the aircraft CG, in meters (m).	0.0
TailRotor_Diameter	Rotor diameter, in m	Required, if Rotor_Radius not defined
TailRotor_Radius	Rotor radius, in m	Required, if Rotor_Diameter not defined
TailRotor_Inertia	Rotor moment of inertia, in kg*m ²	0.001

TailRotor_GearRatio	Engine-to-rotor gear ratio	1.0
TailRotor_Sense	Rotor sense of rotation (set to 1 for positive X-axis rotation, or to -1 for negative X-axis rotation). This controls the sense of the tail rotor thrust force.	1
TailRotor_NumberOfBlades	Number of blades	0
TailRotor_BladeLiftSlope	Slope of the coefficient of lift for the blade airfoil, per rad.	2π
TailRotor_BladeProfileDrag	Profile drag coefficient of the blade airfoil	0.0
TailRotor_BladeChord	Blade chord, in m	0.0
TailRotor_IsVariableCollective	1 if rotor has variable collective controlled by a servo, or 0 if the rotor collective pitch angle is fixed	0
TailRotor_CollectivePitch	If rotor collective is fixed, this defines the blade pitch angle, in deg.	0.0
TailRotor_CollectiveChannel	If rotor has variable collective, this defines the collective input servo channel	3 for left rotor, 8 for right rotor
TailRotor_GyroGain	Proportional gain for yaw rate feedback to tail rotor collective, in rad/(rad/s)	0.0

6.2.4.1 Helicopter Combo Main Rotor + Tail Rotor

The helicopter combo unit models a typical helicopter configuration in which the main rotor and the tail rotor are driven by the same engine, but with different gear ratios. This model makes use of the main rotor and the tail rotor models discussed in the previous subsections. The aircraft configuration file should define the parameters of the main rotor and of the tail rotor. The following example is a configuration for the Maxi-Joker 2 helicopter:

// Main rotor + tail rotor system

Left_Actuator_Type=4

// Main rotor definition

Left_MainRotor_X=0.0

Left_MainRotor_Y=0.0

Left_MainRotor_Z=-0.335

Left_MainRotor_Radius=0.8754

Left_MainRotor_GearRatio=9.25

Left_MainRotor_Sense=-1

Left_MainRotor_NumberOfBlades=2

Left_MainRotor_BladeLiftSlope=5.44

Left_MainRotor_BladeProfileDrag=0.0135

Left_MainRotor_BladeChord=0.0593

Left_MainRotor_GroundEffect=0.05

Left_MainRotor_IsVariableCollective=1

Left_MainRotor_BladeHingeOffset=0.0522

Left_MainRotor_BladeMass=0.230

Left_MainRotor_BladeTwist=0.0

Left_MainRotor_TeeterDamping=0.0

Left_MainRotor_MaxTPPTilt=20

Left_MainRotor_MaxAdvanceRatio=0.2

Left_MainRotor_FlybarRadius=0.3415

Left_MainRotor_FlybarLiftSlope=3.0

Left_MainRotor_FlybarBladeChord=0.045

Left_MainRotor_FlybarBladeSpan=0.128

Left_MainRotor_FlybarBladeInertia=0.00412

Left_MainRotor_FlybarCyclicGain=3.8177

Left_MainRotor_CyclicFlybarTPPGain=0.6529

Left_MainRotor_SwashplateType=5

Left_MainRotor_Servo1Channel=0

Left_MainRotor_Servo2Channel=1

Left_MainRotor_Servo3Channel=2

Left_MainRotor_CyclicSwashplateGain=0.422

Left_MainRotor_BladeGripArm=3.415

Left_MainRotor_SwashplateRadius=3.522

// Tail rotor definition

Left_TailRotor_X=-1.065

Left_TailRotor_Y=0.057

Left_TailRotor_Z=-0.16

Left_TailRotor_Radius=0.1725

Left_TailRotor_GearRatio=2.2

Left_TailRotor_Sense=1

Left_TailRotor_NumberOfBlades=2

Left_TailRotor_BladeLiftSlope=3.2

Left_TailRotor_BladeProfileDrag=0.01

Left_TailRotor_BladeChord=0.0295

Left_TailRotor_IsVariableCollective=1

Left_TailRotor_CollectiveChannel=3

Left_TailRotor_GyroGain=0.1

7 Landing Gear Model

The landing gear model provides the forces and moments that are applied to the airframe when a specific set of points on the aircraft come in contact with the ground. This includes not just the landing gear itself, but also other parts of the airframe that may come in contact with the ground. The ground contact points are each modeled with coefficients for stiffness, damping, and longitudinal/lateral friction. There are two types of contact points: simple, and wheels. They differ through the coefficients listed above. The longitudinal friction of a wheel is actually a roll-friction type (resisting force proportional to the velocity). The wheels can also steer – the friction forces are computed in the local frame of the wheel and rotated back to the aircraft frame.

The user can specify the location (with respect to the aircraft CG) of three wheels and nine other contact points. The entire set is represented in the following two figures.

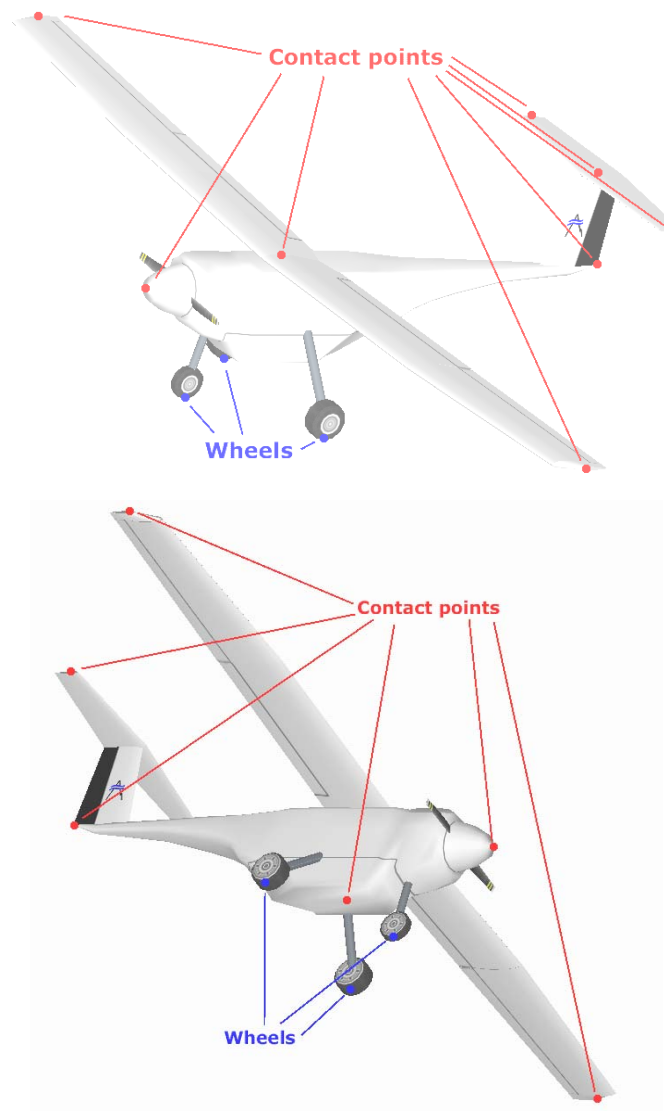


Figure 24 - Top and Bottom View of Ground Contact Points and Wheels

Table 12 - Wheel Contact Points

Value	Meaning	Default
NoseWheel_Position_X	The X-axis component of the position vector of the nose wheel or tail wheel ground contact point with respect to the aircraft CG, in m. (positive if ahead of CG)	0.0
NoseWheel_Position_Y	The Y-axis component of the position vector of the nose wheel or tail wheel ground contact point with respect to the aircraft CG, in m. (positive to the right)	0.0
NoseWheel_Position_Z	The Z-axis component of the position vector of the nose wheel or tail wheel ground contact point with respect to the aircraft CG, in m. (positive if below the CG)	0.0
NoseWheel_RudderWheelRatio	If nose wheel is mechanically linked to the rudder, this is the ratio of steering angle to rudder deflection angle (negative for nose wheel, positive for tail wheel, leave at 0.0 for non-steerable)	0.0
NoseWheel_Steering_Channel	The Piccolo servo channel number for the steering actuator.	0
RightWheel_Position_X	The X-axis component of the position vector of the right wheel of the main gear with respect to the aircraft CG, in m. (positive if ahead of CG)	0.0
RightWheel_Position_Y	The Y-axis component of the position vector of the right wheel of the main gear with respect to the aircraft CG, in m. (positive to the right)	0.0
RightWheel_Position_Z	The Z-axis component of the position vector of the right wheel of the main gear with respect to the aircraft CG, in m. (positive if below the CG)	0.0
LeftWheel_Position_X	The X-axis component of the position vector of the left wheel of the main gear with respect to the aircraft CG, in m. (positive if ahead of CG)	0.0
LeftWheel_Position_Y	The Y-axis component of the position vector of the left wheel of the main gear with respect to the aircraft CG, in m. (positive to the right)	0.0
LeftWheel_Position_Z	The Z-axis component of the position vector of the right wheel of the main gear with respect to the aircraft CG, in m. (positive if below the CG)	0.0

Table 13 - Other Contact Points

Value	Meaning	Default
ContactPoint_Top_Position_X	The X-axis component of the position vector of the top-mid fuselage point w.r.t aircraft CG, in m.	0.0
ContactPoint_Top_Position_Y	The Y-axis component of the position vector of the top-mid fuselage point w.r.t aircraft CG, in m.	0.0
ContactPoint_Top_Position_Z	The Z-axis component of the position vector of the top-mid fuselage point w.r.t aircraft CG, in m.	0.0
ContactPoint_Bottom_Position_X	The X-axis component of the position vector of the bottom-mid fuselage point w.r.t aircraft CG, in m.	0.0
ContactPoint_Bottom_Position_Y	The Y-axis component of the position vector of the bottom-mid fuselage point w.r.t aircraft CG, in m.	0.0
ContactPoint_Bottom_Position_Z	The Z-axis component of the position vector of the bottom-mid fuselage point w.r.t aircraft CG, in m.	0.0

ContactPoint_Nose_Position_X	The X-axis component of the position vector of the fuselage nose point w.r.t aircraft CG, in m.	0.0
ContactPoint_Nose_Position_Y	The Y-axis component of the position vector of the fuselage nose point w.r.t aircraft CG, in m.	0.0
ContactPoint_Nose_Position_Z	The Z-axis component of the position vector of the fuselage nose point w.r.t aircraft CG, in m.	0.0
ContactPoint_Tail_Position_X	The X-axis component of the position vector of the fuselage tail endpoint w.r.t aircraft CG, in m.	0.0
ContactPoint_Tail_Position_Y	The Y-axis component of the position vector of the fuselage tail endpoint w.r.t aircraft CG, in m.	0.0
ContactPoint_Tail_Position_Z	The Z-axis component of the position vector of the fuselage tail endpoint w.r.t aircraft CG, in m.	0.0
ContactPoint_LWing_Position_X	The X-axis component of the position vector of the left wingtip w.r.t aircraft CG, in m.	0.0
ContactPoint_LWing_Position_Y	The Y-axis component of the position vector of the left wingtip w.r.t aircraft CG, in m.	0.0
ContactPoint_LWing_Position_Z	The Z-axis component of the position vector of the left wingtip w.r.t aircraft CG, in m.	0.0
ContactPoint_RWing_Position_X	The X-axis component of the position vector of the right wingtip w.r.t aircraft CG, in m.	0.0
ContactPoint_RWing_Position_Y	The Y-axis component of the position vector of the right wingtip w.r.t aircraft CG, in m.	0.0
ContactPoint_RWing_Position_Z	The Z-axis component of the position vector of the right wingtip w.r.t aircraft CG, in m.	0.0
ContactPoint_LStab_Position_X	The X-axis component of the position vector of the left horizontal stabilizer tip w.r.t aircraft CG, in m.	0.0
ContactPoint_LStab_Position_Y	The Y-axis component of the position vector of the left horizontal stabilizer tip w.r.t aircraft CG, in m.	0.0
ContactPoint_LStab_Position_Z	The Z-axis component of the position vector of the left horizontal stabilizer tip w.r.t aircraft CG, in m.	0.0
ContactPoint_RStab_Position_X	The X-axis component of the position vector of the right stabilizer tip w.r.t aircraft CG, in m.	0.0
ContactPoint_RStab_Position_Y	The Y-axis component of the position vector of the right stabilizer tip w.r.t aircraft CG, in m.	0.0
ContactPoint_RStab_Position_Z	The Z-axis component of the position vector of the right stabilizer tip w.r.t aircraft CG, in m.	0.0
ContactPoint_Fin_Position_X	The X-axis component of the position vector of the top of the vertical tail w.r.t aircraft CG, in m.	0.0
ContactPoint_Fin_Position_Y	The Y-axis component of the position vector of the top of the vertical tail w.r.t aircraft CG, in m.	0.0
ContactPoint_Fin_Position_Z	The Z-axis component of the position vector of the top of the vertical tail w.r.t aircraft CG, in m.	0.0

8 Sensor Models

The Simulator supports sensor models that can be used to alter the sensor information sent to the avionics. The sensor data are included in a separate file, which is referenced from the main file using the flag `Sensors=` followed by the path name to the sensors file. The sensors that are modeled are given in **Table 14**.

Table 14 - Sensor Names For Simulator Sensor Model

Value	Meaning
Latitude_Sensor_	Latitude of the vehicle from the GPS
Longitude_Sensor_	Longitude of the vehicle from the GPS
Height_Sensor_	Height of the vehicle from the GPS
VNorth_Sensor_	North component of ground speed from the GPS
VEast_Sensor_	East component of ground speed from the GPS
VDown_Sensor_	Down component of ground speed from the GPS
PDynamic_Sensor_	Dynamic pressure sensor
PStatic_Sensor_	Static pressure sensor
Roll_Rate_Sensor_	Avionics axis roll rate sensor
Pitch_Rate_Sensor_	Avionics axis pitch rate sensor
Yaw_Rate_Sensor_	Avionics axis yaw rate sensor
X_Accel_Sensor_	Avionics x-axis acceleration sensor
Y_Accel_Sensor_	Avionics y-axis acceleration sensor
Z_Accel_Sensor_	Avionics z-axis acceleration sensor

Each parameter in the file is formed by concatenating the sensor name with the parameter name. Each sensor can be modeled according to the parameters in **Table 15**.

Table 15 - Sensor Modeling Parameters

Value	Meaning	Units
Offset	Sensor output when the signal being sensed is at zero	Output
Order	Order of the butterworth low pass filter that the sensor output is passed through. Use 0 for no filter	N/A
Bandwidth	Cutoff frequency in Hz of the low pass filter.	Hz
Gain	Gain of the sensor signal.	N/A
Resolution	Resolution of the sensor signal	Output
Noise	Sensor noise, used to scale random number added to the sensor output before going through the low pass filter	Output
Min	Minimum saturation limit.	Output
Max	Maximum saturation limit.	Output
Drift_Rate	Maximum offset drift rate.	Output/s
Max_Drift	Maximum offset drift value.	Output
Drift_Hold	Amount of time to spend at a specific offset drift value before allowing the sensor to drift again.	s

In addition to the individual sensor parameters, there are also global sensor parameters controlling the GPS update rates (**Table 16**).

Table 16 - Sensor Modeling Parameters

Value	Meaning	Units
GPS_Period	The update rate of the GPS. This value should be 1000 for the M12 GPS and 250 for the uBlox GPS (used in Piccolo II)	ms
GPS_Position_Lag	The time lag of the position output, prior to the velocity projection. This should be 500 for the M12 and 125 for the uBlox	ms
GPS_Velocity_Lag	The time lag of the velocity output. Should be 1100 for the M12 and 250 for the uBlox.	ms

The Sensor models can be loaded, saved, or modified from the Actuator interface in the Simulator. **SensorPerfect** and **SensorPoor** are the two default sensor models that are provided in the Piccolo Developer's Kit.

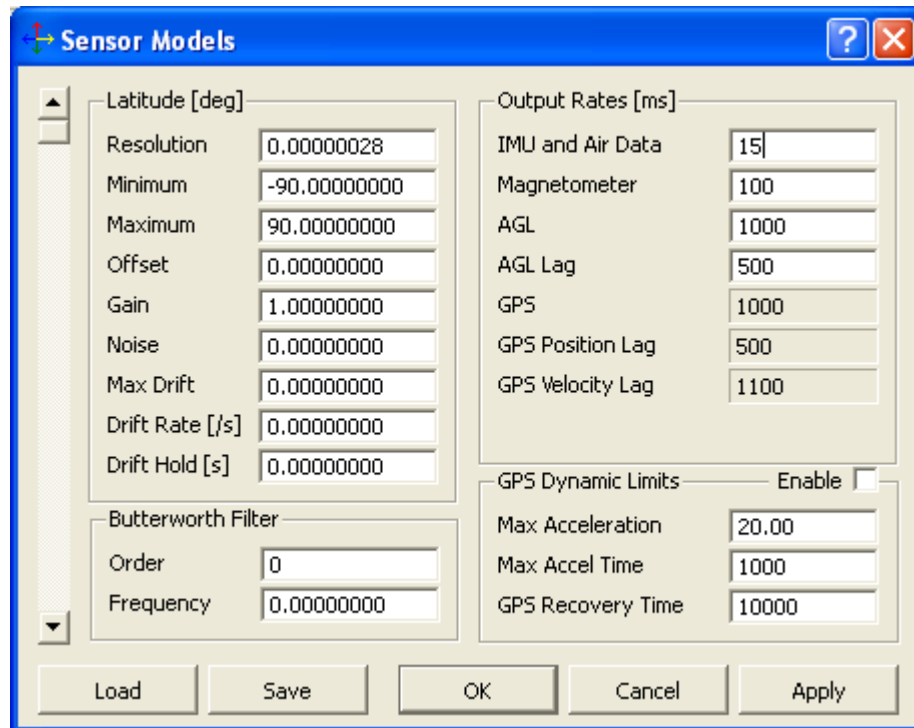


Figure 25 - Sensor Models

9 Actuator Models

The Simulator supports actuator models that can be used to limit the responsiveness of the control surface actuators. The actuator data are included in a separate file, which is referenced from the main file using the flag `Actuators=` followed by the path name to the actuators file. The actuators that are modeled are shown in **Table 17**.

Table 17 - Actuator Names for Simulator Actuators Model

Actuator	Channel number	Output units
Left_Aileron_	0	Radians
Left_Elevator_	1	Radians
Left_Throttle_	2	0-1
Left_Rudder_	3	Radians
Left_Flap_	4	Radians
Right_Aileron_	5	Radians
Right_Elevator_	6	Radians
Right_Throttle_	7	0-1
Right_Rudder_	8	Radians
Right_Flap_	9	Radians

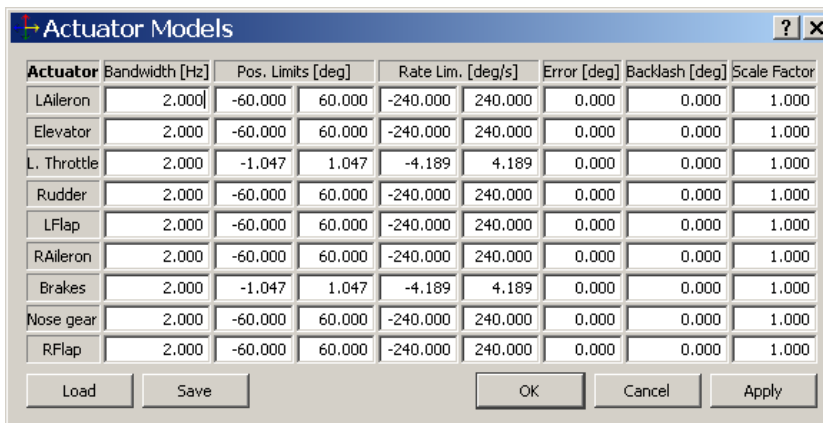
Each parameter in the file is formed by concatenating the actuator name with the parameter name. The actuators can be modeled according to the parameters in **Figure 18**.

Table 18 - Actuator Modeling Parameters

Value	Meaning	Units	Defaults
Bandwidth	Bandwidth of the actuator	Hz	2
Rate_Limit	Maximum rate of actuator	output/s	4.189
Min_Limit	Minimum deflection of the actuator	output	-1
Max_Limit	Maximum deflection of the actuators	output	1
Error	Output error of the actuator	output	0
Backlash	Backlash error of the actuator	output	0
Scale Factor	Multiplier	NA	1

Each actuator model in the Simulator includes a 2nd order bandwidth limiter, position limiter, rate limiter, backlash, and Scale Factor.

The actuator models can be loaded, saved, or modified from the Actuator interface in the Simulator. The following five actuator models are provided in the Piccolo Developers Kit: Fast, Sloppy, Slow, Standard, and Very Slow.



The dialog box titled "Actuator Models" contains a table with the following data:

Actuator	Bandwidth [Hz]	Pos. Limits [deg]		Rate Lim. [deg/s]		Error [deg]	Backlash [deg]	Scale Factor
LAileron	2.000	-60.000	60.000	-240.000	240.000	0.000	0.000	1.000
Elevator	2.000	-60.000	60.000	-240.000	240.000	0.000	0.000	1.000
L. Throttle	2.000	-1.047	1.047	-4.189	4.189	0.000	0.000	1.000
Rudder	2.000	-60.000	60.000	-240.000	240.000	0.000	0.000	1.000
LFlap	2.000	-60.000	60.000	-240.000	240.000	0.000	0.000	1.000
RAileron	2.000	-60.000	60.000	-240.000	240.000	0.000	0.000	1.000
Brakes	2.000	-1.047	1.047	-4.189	4.189	0.000	0.000	1.000
Nose gear	2.000	-60.000	60.000	-240.000	240.000	0.000	0.000	1.000
RFlap	2.000	-60.000	60.000	-240.000	240.000	0.000	0.000	1.000

Buttons at the bottom: Load, Save, OK, Cancel, Apply.

Figure 26 - Actuator Models

10 Launcher Model

The Simulator supports a rail launcher. The rail launcher simulation is intended to model simple catapult launch systems where the vehicle is accelerated along a fixed rail. The rail length, force, heading, and pitch angle can be controlled with parameters in the dynamics file.

Table 19 - Launcher Parameters Values

Value	Meaning	Defaults
Launcher_Length	The length of the launch rail in meters	10.0
Launcher_Initial_Force	The force applied to the vehicle in the direction of the rail, at the start of the launch	160.0
Launcher_Final_Force	The force applied to the vehicle in the direction of the rail, at the end of the launch. Between the beginning and the end of the launch the force applied is scaled linearly from the initial force to the final force	160.0
Launcher_Heading	The heading of the launch rail in degrees. This determines the direction the vehicle will move as it accelerates on the rail	0.0
Launcher_Pitch	The pitch angle of the launch rail in degrees. This determines how steeply the vehicle will be climbing when it leaves the rail.	20.0

11 Initialization

After a model is loaded the Simulator will initialize all the state variables with zero. The initialization can be overridden using inputs available on the screen. The state of the model can be saved to a file and loaded later as initialization data. The form of the file is similar to the dynamics model files, i.e each variable is named in the file. A list of the state variables that can be initialized is shown in **Table 20**.

The State file can be loaded or saved from the Simulator interface from the **File » Load State...** or **File » Save State...**

Table 20 - State Initialization File

Value	Meaning
Alpha	Angle of attack of the model, in degrees
Beta	Angle of sideslip of the model, in degrees
Roll	Euler roll angle, in degrees
Pitch	Euler pitch angle, in degrees
Yaw	Euler heading angle, in degrees
P	Body axis roll rate, in degrees per second
Q	Body axis pitch rate, in degrees per second
R	Body axis yaw rate, in degrees per second
TAS	True air speed in meters per second
Latitude	Latitude in degrees
Longitude	Longitude in degrees
Altitude	Altitude in meters
Left_Engine_RPM	RPM of the left engine
Right_Engine_RPM	RPM of the right engine

12 Piccolo Parameters

Piccolo flight control laws are based on the aircraft parameters. For typical applications, once the aerodynamic parameters are defined, the default set of gains should fly the aircraft. The Simulator has the ability to generate an aircraft parameters file from the Simulator input file. From the Simulator menu select **Vehicle Data » Fixed Wing Gen 2**. This will create an XML file with all the vehicle parameters defined in the **Vehicle** tab of the **Controller Configuration** window in PCC (**Figure 27**).

Controller Configuration - Piccolo '1'

Lat Gains | Lon Gains | Trims | Limits | **Vehicle** | Mixing | Landing | Launch | Sensor Navigation

Open... Save... Request All Send All Set to Defaults

<p>Flags</p> <p>Auto elevator effect <input type="text" value="0"/></p> <p>Auto aileron effect <input type="text" value="0"/></p>	<p>Lateral aero</p> <p>Aileron effect <input type="text" value="0.62267"/> Cp/da</p> <p>Rudder power <input type="text" value="0.000858"/> [/deg]</p> <p>Rudder effect <input type="text" value="-0.802141"/> B/dr</p> <p>Sideslip effect <input type="text" value="-0.005647"/> [/deg]</p>
<p>Geometry</p> <p>Wing area <input type="text" value="0.939"/> [m²]</p> <p>Wing span <input type="text" value="3.04"/> [m]</p> <p>Vertical tail arm <input type="text" value="0.00"/> [m]</p> <p>Steering arm <input type="text" value="0.30"/> [m]</p>	<p>Engine</p> <p>Max engine power <input type="text" value="1000"/> [W]</p> <p>Engine SFC <input type="text" value="600"/> [g/(kW-hr)]</p>
<p>Mass properties</p> <p>Gross Mass <input type="text" value="18.000"/> [kg]</p> <p>Empty Mass <input type="text" value="15.000"/> [kg]</p> <p>X Inertia <input type="text" value="2.700"/> [kg-m²]</p> <p>Y Inertia <input type="text" value="3.200"/> [kg-m²]</p> <p>Z Inertia <input type="text" value="5.700"/> [kg-m²]</p> <p>Payload Mass <input type="text" value="0.000"/> [kg]</p>	<p>Lift coefficients</p> <p>CL max <input type="text" value="1.200"/></p> <p>CL climb <input type="text" value="0.900"/></p> <p>CL cruise <input type="text" value="0.750"/></p>
<p>Longitudinal aero</p> <p>Elevator power <input type="text" value="-0.025617"/> [/deg]</p> <p>CL at zero elevator <input type="text" value="0.444569"/> CL</p> <p>Elevator effect <input type="text" value="-0.143522"/> [/deg]</p>	

Figure 27 - Controller Configuration

13 Installation and Operation

The Simulator works as a standalone executable. No special installation is required, however the CAN interface driver must be installed. This is done using the USBCanModule driver disc. In addition when starting the Simulator the USB to CAN module must be plugged in. The Simulator will detect that it is plugged in and configures itself to use it as its source of control surface data. If the CAN module is not connected, the Simulator will attempt to connect to a software simulation source. If it fails to connect to a software simulation source, it will use any installed joysticks for the control surface data.

13.1 Installing FlightGear

The current version of the Simulator supports FlightGear versions 0.9.2, 0.9.4, 0.9.8, 0.9.9, and 0.9.10.

13.1.1 FlightGear Version 0.9.2

FlightGear version 0.9.2 should be installed by unzipping its archives. First, make sure that the directory C:\FlightGear on your computer does not exist already. If it exists and you would like to keep that version, rename that directory. Otherwise, delete it before installing the newer version.

- Unzip the file **fgfs-base-0.9.2.zip** to the root directory of your C: drive. This will create a new directory called FlightGear and its directory tree. This archive contains most of the FlightGear data.
- Unzip the file **fgfs-win32-msvc-bin-0.9.2.zip** to the root directory of your C: drive. This will add more files to the FlightGear directory and its tree. This archive contains the binary executables of FlightGear.
- Copy the two batch files **runfgfsnet-c172.bat** and **runfgfsnet-j3cub.bat** to the C:\FlightGear directory. These two batch files can be used to launch the FlightGear program in "external flight dynamics" mode, which is exactly what we want for interfacing with our Simulator. The only difference between the two files is that they load different visual models - one of them loads a Cessna 172, and the other one a Piper Cub.
- Optionally, unzip the files **wXXXnXX.tar.gz** to C:\FlightGear\Data\Scenery. The four files included provide scenery and airports for the west coast of the United States.

13.1.2 FlightGear Version 0.9.4 / 0.9.8 / 0.9.9 / 0.9.10

The newer versions of FlightGear for Windows platform come as a self-installing archive. Run the **fgsetup0.9.4.exe** program. The Setup will guide you through the installation. When completed, copy the two batch files **runfgfsnet-c172.bat** and **runfgfsnet-j3cub.bat** to the directory where FlightGear was installed. These two batch files can be used to launch the FlightGear in the correct mode.

13.2 Starting Simulator

1. Start FlightGear by using one of the supplied batch files. This will start FlightGear in external flight dynamics mode. The program accepts aircraft state data (position, velocity and attitude) from a UDP network connection on port 5500. If you prefer to use the default start batch file, the command line for launching FlightGear in external dynamics mode is:

```
runfgfs.bat --native-fdm=socket,in,30,,5500,udp --fdm=external
```

2. On the Simulator computer, launch the Simulator application. Select **File » Open** to open the text file that contains the simulation model.
3. From the top menu, select **External/FlightGear...** Type the network name of the computer running FlightGear in the dialogue box. The aircraft in FlightGear should now be exactly at the location that it is set in the Simulator.

4. The aircraft states can now be initialized properly in the Simulator, after which you can start the simulation anytime by pressing the Start button. The **stop** button halts the simulation but maintains the aircraft states to the last computed values. The **Reset** button resets the aircraft states to the program defaults.

13.3 FlightGear View Options

This section covers the view options in FlightGear. The Cockpit View (**Figure 28**) is the first view shown upon startup.



Figure 28 - FlightGear Cockpit View

Most of the instruments in the cockpit view are not useful for a UAV simulation since they are designed for manned aircraft and are not connected to any external flight dynamics model. To remove the cockpit clutter use the *S* key (**Figure 29**). The cockpit can be enabled by pressing the *S* key again.

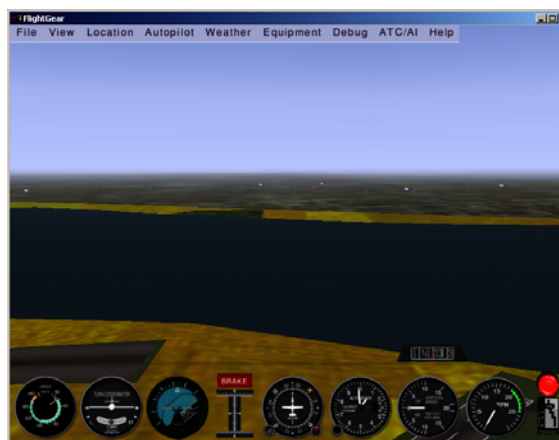


Figure 29 - Forward View Cockpit Removed

The forward-looking pilot view is one of multiple views available in FlightGear. Views are fully configurable, so any view is theoretically possible. For more information about configuring views, see the [FlightGear viewer document](#).

To navigate between different views use the *V* key. There is a set of default views that are already configured when FlightGear is started. There are also two external views: fixed orientation with respect to the airplane, and fixed orientation with respect to the ground (inertial frame). An example of the external view is shown in **Figure 30**.



Figure 30 - External View

For the external views, change the location of the camera with respect to the airplane using the *SHIFT* + arrow keys. The zoom can be adjusted using the *X* key (zoom in) and *SHIFT*+*X* keys (zoom out).

Another type of view is the ground/tower view **Figure 31**. This is particularly useful for flying UAVs/RPVs under manual control.



Figure 31 - Tower View

13.4 Displaying Multiple Aircraft

If there are multiple Piccolo Simulators running on the local network, all the aircraft can be displayed in a single FlightGear window **Figure 32**. Each instance of the Simulator application should be connected to individual FlightGear computers, but with the appropriate command line parameters. On the client FlightGear computer, run FlightGear with the following command line parameters:

- `runfgfs.bat --aircraft=j3cub --native-fdm=socket,in,30,,5500,udp`
- `--fdm=external --multiplay=out,10,XXX.XXX.XXX.XXX,5501 --allsign=player1`

On the server FlightGear computer (where all aircraft will be displayed) run FlightGear with the following command line parameters:

- `runfgfs.bat --aircraft=j3cub --native-fdm=socket,in,30,,5500,udp`
- `--fdm=external --multiplay=in,10,XXX.XXX.XXX.XXX,5501 --callsign=player2`

***Note:** XXX.XXX.XXX.XXX is the IP address of the server.*



Figure 32 - Multiple Aircraft

13.5 Displaying Custom 3-D Aircraft Models

FlightGear can accept models in various 3-D formats. For more information regarding aircraft modeling see the [FlightGear modeling documentation](#).

In this example, we will use the Predator FS2000 model.

1. Copy the predator.md 3-D model file to the C:\FlightGear\data\Models\ directory.
2. In the same directory create a small XML file that defines the path property to the 3-D model file. For this example, the file should contain the following:
 - `<?xml version="1.0"?>`
 - `<PropertyList>`
 - `<path>Models/predator.mdl</path>`
 - `</PropertyList>`
3. Save the file as **predator.xml**.
4. In the batch file that was used to start FlightGear, add a new command line parameter: `--prop:/sim/model/path=`. This path points to the newly created predator.xml file. The new batch file should contain the following:
 - `%TOP_ROOT%\BIN\WIN32\FGFS.EXE`
 - `--prop:/sim/model/path=%FG_ROOT%\Models\predator.xml`
 - `--native-fdm=socket,in,30,,5500,udp --fdm=external`
5. Launching FlightGear with this new batch file will result in loading the Predator FS2000 model as shown in **Figure 33**.



Figure 33 - Predator FS2000 Model

14 Appendix

14.1 Inertia Data

Some of these parameters may be omitted or set to zero (many of them default to 0), but all 3 axes must have non-zero inertias or the Simulator will declare it an invalid model. An inertia spreadsheet is available (included in the AVL archive) which does a better job at modeling V-tail aircraft, and allows you to interactively adjust the model. It also calculates CG location from components.

Gross_Mass=	Fuse_Z=
Empty_Mass=	Tail_Mass=
Left_Engine_Mass=	Tail_Area=
Left_Prop_X=	Tail_Span=
Left_Prop_Y=	Tail_X=
Left_Prop_Z=	Tail_Z=
Right_Engine_Mass=	Left_Fin_Mass=
Right_Prop_X=	Left_Fin_Area=
Right_Prop_Y=	Left_Fin_Span=
Right_Prop_Z=	Left_Fin_X=
	Left_Fin_Y=
	Left_Fin_Z=
Wing_Mass=	Right_Fin_Mass=
Wing_Area=	Right_Fin_Area=
Wing_Span=	Right_Fin_Span=
Wing_X=	Right_Fin_X=
Wing_Z=	Right_Fin_Y=
Fuselage_Mass=	Right_Fin_Z=
Fuse_X=	

14.2 Stability Derivative List

These are the stability derivatives that are used by the Simulator (and are output by AVL). This ability to import XML data could be the basis for wind tunnel data support in the CCT Simulator.

Note: The drag used by the current AVL Simulator aerodynamics module is $CD_{ff} + CD_{vis}$.

All of these derivatives can be tables vs. angle of attack.

Label	Description
CDff	Trefftz plane induced drag coefficient
CDvis	Profile drag + CDo drag coefficient
CY	Side force coefficient
CL	Lift coefficient
CI	Roll moment coefficient
Cm	Pitch moment coefficient
Cn	Yaw moment coefficient
CLb	Variation of lift coefficient with sideslip angle (or use beta sweep)
CYb	Variation of side force coefficient with sideslip angle (or use beta sweep)
CIb	Variation of roll moment coefficient with sideslip angle (or use beta sweep)
Cmb	Variation of pitch moment coefficient with sideslip angle (or use beta sweep)
Cnb	Variation of yaw moment coefficient with sideslip angle (or use beta sweep)

CLp	Variation of lift coefficient with roll rate
CYp	Variation of side force coefficient with roll rate
Clp	Variation of roll moment coefficient with roll rate
Cmp	Variation of pitch moment coefficient with roll rate
Cnp	Variation of yaw moment coefficient with roll rate
CLq	Variation of lift coefficient with pitch rate
CYq	Variation of side force coefficient with pitch rate
Clq	Variation of roll moment coefficient with pitch rate
Cmq	Variation of pitch moment coefficient with pitch rate
Cnq	Variation of yaw moment coefficient with pitch rate
CLr	Variation of lift coefficient with yaw rate
CYr	Variation of side force coefficient with yaw rate
Clr	Variation of roll moment coefficient with yaw rate
Cmr	Variation of pitch moment coefficient with yaw rate
Cnr	Variation of yaw moment coefficient with yaw rate
CDffd1	Variation of induced drag coefficient with control surface 1 deflection (deg) (note replace 1 with control surface # 1..n for this and all other d1 derivatives)
CLd1	Variation of lift coefficient with control surface 1 deflection (deg)
CYd1	Variation of side force coefficient with control surface 1 deflection (deg)
CLd1	Variation of roll moment coefficient with control surface 1 deflection (deg)
Cmd1	Variation of pitch moment coefficient with control surface 1 deflection (deg)
Cnd1	Variation of yaw moment coefficient with control surface 1 deflection (deg)

14.3 Prop Example File

Sample PRD file for an APC 20 x 8 propeller.

```
# Prop: 20x8
# RPM 4700 [m/s]
# Number of panels 0
# J Cp Ct %
-1.0000 0.0200 0.0600
0.2000 0.0250 0.0515
0.2441 0.0248 0.0508
0.3000 0.0245 0.0478
0.3800 0.0234 0.0422
0.4386 0.0216 0.0352
0.5000 0.0187 0.0262
0.5414 0.0160 0.0196
0.5883 0.0122 0.0110
0.6000 0.0111 0.0085
0.6083 0.0103 0.0068
0.6221 0.0089 0.0029
0.6331 0.0078 0.0000
0.7000 0.0000 -0.0050
0.8000 -0.0050 -0.0156
0.9000 -0.0097 -0.0203
1.0000 -0.0180 -0.0295
1.2000 -0.0273 -0.0400
2.0000 -0.0737 -0.1115
```