AVL Tool

Getting started

Required files

AVL Tool requires the AVL executable to be in the same folder as AVL Tool.
Place all *.avl*, *.mass*, *.run*, and airfoil files in the folder with AVL and AVL Tool.
Any other files needed by AVL Tool will be generated when using the application.

Included in the wrapper package is the *vanilla.avl* plane and its *sd7037.dat* airfoil file. These are used for the tutorials included in this guide.

To use your own *.avl* files and airfoil files, place them into the same folder as AVL and AVL Tool.

Required experience

AVL Tool does not require any programming knowledge or IDE. Users will greatly benefit from understanding AVL geometry and how to use AVL, but this experience is not entirely required to follow the tutorials.

However, the user will need to have a basic understanding of aerodynamics and aircraft force coefficients to avoid being completely lost.

This document will walk the user through performing analysis using AVL Tool, including editing aircraft geometry, calculating the aircraft trim state, and creating an inviscid drag polar.

Operating system note

AVL Tool has only been tested on Windows 10 and Windows 11 operating systems. The library used to make the GUI, Dear ImGui, is cross-platform; however, AVL Tool uses Windows-specific functions to create and manage child processes (AVL).

Obligatory tour

Folder contents

To best use AVL Tool, you should place your .avl file and airfoil files into the same folder as AVL and AVL Tool. Your .avl files should be written to use airfoils in the same folder.

AVL Tool creates a few files to help with wrapper business:

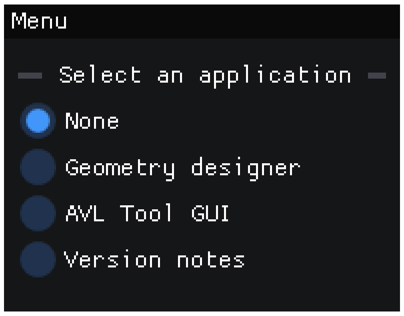
- **_(output type).AVLOUTPUT** files are AVL-generated output files.
- **OUTPUT.dat** contains the user-specified output values from AVL.
- **input_list.txt** is the set of commands fed directly into AVL.
- **WrapperTicket.ticket** is the set of commands to create one or more **input_list.txt** files

The user does not need to edit these files manually; they are generated by AVL Tool upon AVL execution.

AVL Tool setup

Open AVL Tool and maximize the application window.

On the left-hand side, there is a panel with radio buttons.



“None” is the default. As you can guess (and see), it displays nothing in the application window.

“Geometry designer” is a geometry rendering tool for viewing and editing AVL geometry (.avl files). This will be the topic of our first tutorial.

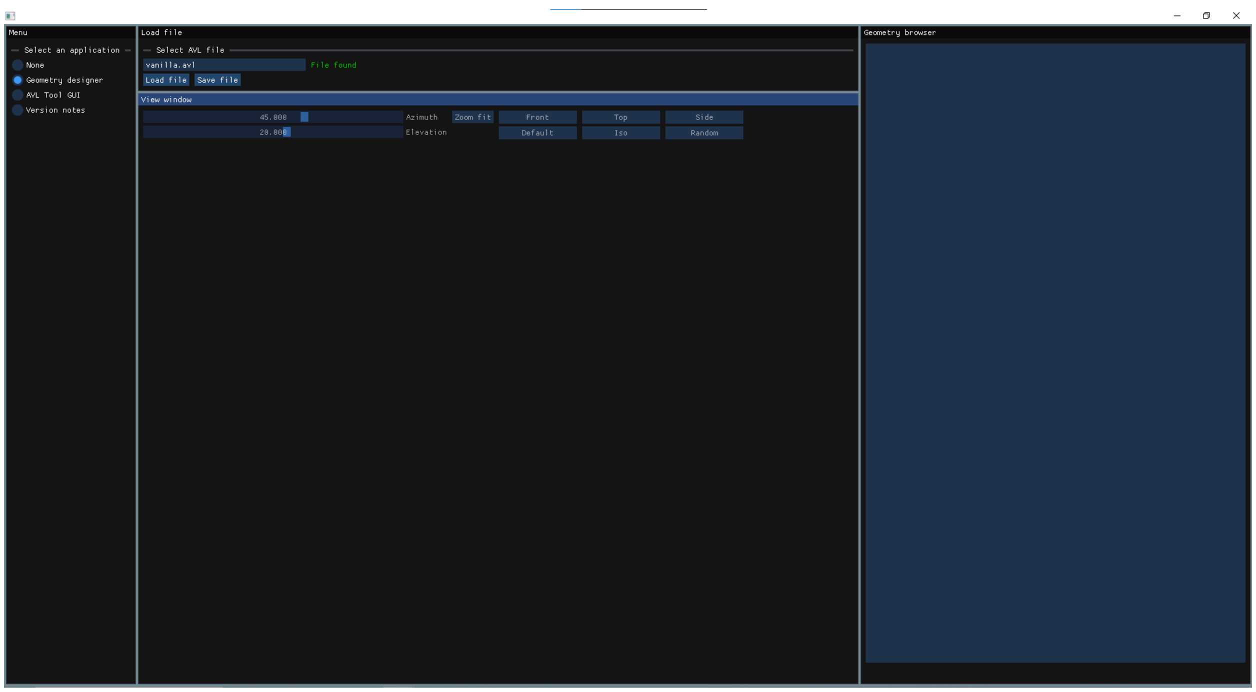
“AVL Tool GUI” is a set of menus used for creating unique run cases for AVL and executing them. This tab is the most complex, and we will focus on it in tutorials 2 and 3.

“Version notes” is unrelated to any aerodynamic analysis, and is for keeping track of changes and new features (otherwise it’d be a text document on my desktop, and I wouldn’t have much incentive to keep updating it). It also serves as something to look back on and think “wow I really did all that huh”. You should also keep a “version notes” for your plane.

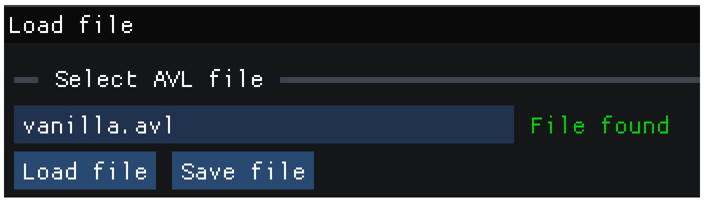
Tutorials

Tutorial 1: Loading, viewing, and modifying geometry.

Open AVL Tool to the “Geometry designer” tab. You will be greeted by a mostly-empty screen.

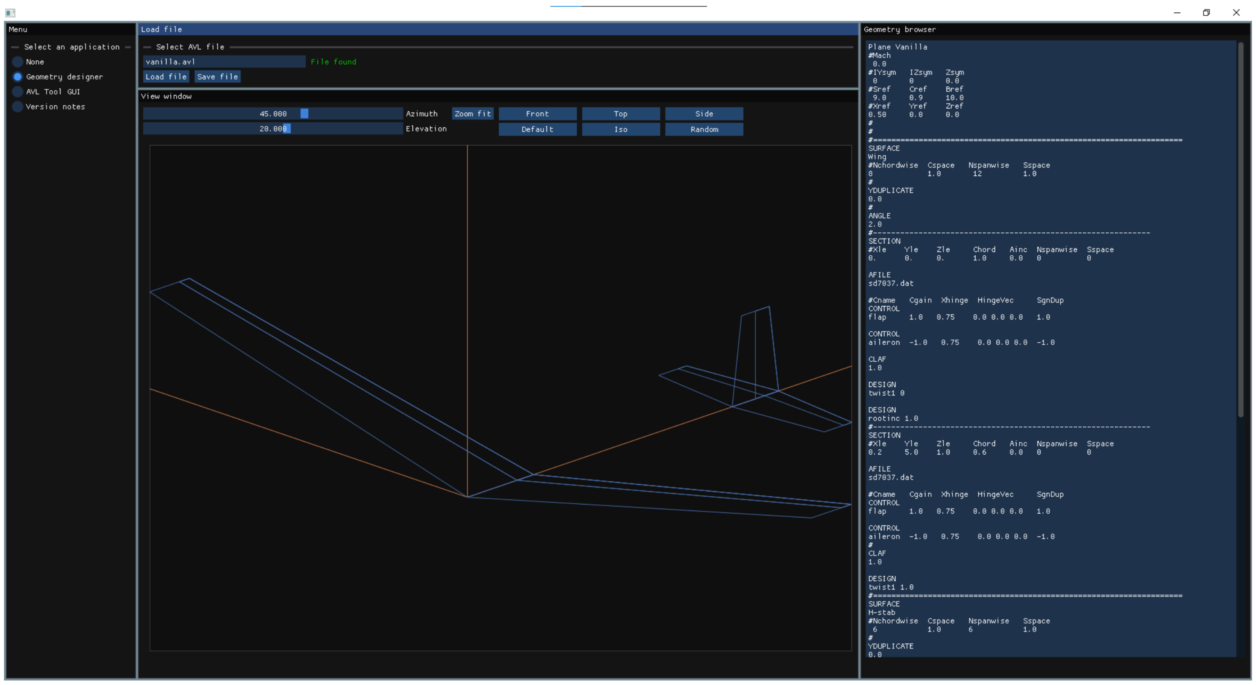


Included in the wrapper package is the “vanilla.avl” plane. This file is written to the text input field by default, and we will use it for our tutorials.



To load a geometry file, enter its filename into the text field. If the file exists in the folder with AVL Tool, you will get a File found. If not found, you’ll get a File not found. And will not be able to load the geometry.

To view your plane, press “Load file”. Your screen should now look like this:



vanilla.avl

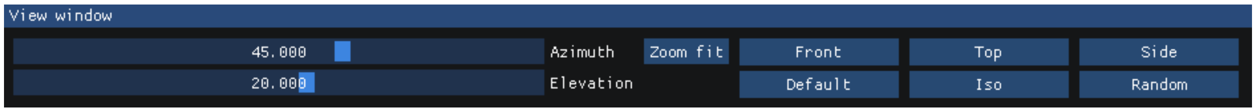
Azimuth 45.000

Elevation 20.000

```
Plane Vanilla
#Mach
 0.0
#IYsym   IZsym   Zsym
 0       0       0.0
#Sref    Cref    Bref
 9.0     0.9     10.0
#Xref    Yref    Zref
0.50     0.0     0.0
#
#
#======================================================================
SURFACE
Wing
#Nchordwise  Cspace   Nspanwise   Sspace
8            1.0      12          1.0
#
YDUPLICATE
0.0
#
ANGLE
2.0
#--------------------------------------------------------------
SECTION
#Xle    Yle    Zle     Chord   Ainc  Nspanwise  Sspace
0.      0.     0.      1.0     0.0   0          0

AFILE
sd7037.dat

#Cname   Cgain  Xhinge  HingeVec     SgnDup
CONTROL
flap     1.0    0.75    0.0 0.0 0.0   1.0

CONTROL
aileron  -1.0   0.75    0.0 0.0 0.0  -1.0

CLAF
1.0

DESIGN
twist1 0

DESIGN
rootinc 1.0
#--------------------------------------------------------------
SECTION
#Xle    Yle    Zle     Chord   Ainc  Nspanwise  Sspace
0.2     5.0    1.0     0.6     0.0   0          0

AFILE
sd7037.dat

#Cname   Cgain  Xhinge  HingeVec     SgnDup
CONTROL
flap     1.0    0.75    0.0 0.0 0.0   1.0

CONTROL
aileron  -1.0   0.75    0.0 0.0 0.0  -1.0
#
CLAF
1.0

DESIGN
twist1 1.0
#======================================================================
SURFACE
H-stab
#Nchordwise  Cspace   Nspanwise   Sspace
 6           1.0      6           1.0
#
YDUPLICATE
0.0
```

The center of the window displays your plane (with control surfaces) and the right-hand text field contains your .avl file contents.

Let’s first try changing the view options.



The view window contains sliders for the azimuth (left-right) and elevation (up-down) angles. You can **drag the sliders to rotate the geometry**, or use **ctrl + left-click on the slider to input a value**.

The “Zoom fit” button resets the display so that the geometry fits the viewing window.

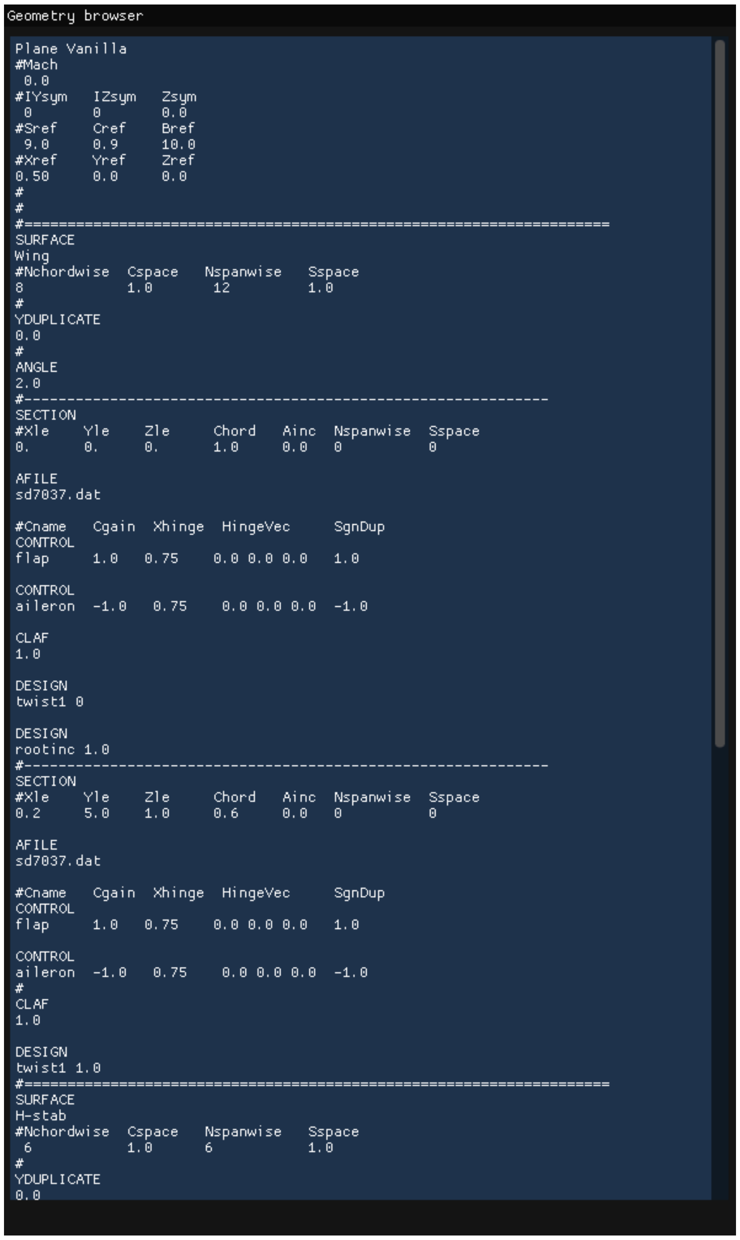
The remaining buttons are preset viewing angles:

- **“Front”** looks down the +X-axis, with the +Z-axis pointing upward. This is what you’d expect from the front of a plane.
- **“Top”** looks down the -Z-axis, with the +X-axis pointing upward.
- **“Side”** views the plane from its left side, down the +Y-axis, with the +Z-axis pointing upward.

- **“Default”** is AVL’s default viewing angle. This is what you’d see if you opened the geometry submenu in the oper menu.
- **“Iso”** is an isometric view. Not much to say about that.
- **“Random”** gives you a randomized view of the plane in case you need inspiration.

On the geometry display, you may **left-click and drag to pan** and **scroll up/down to zoom in/out.**

After trying all those buttons out, let’s check out the right-hand text.

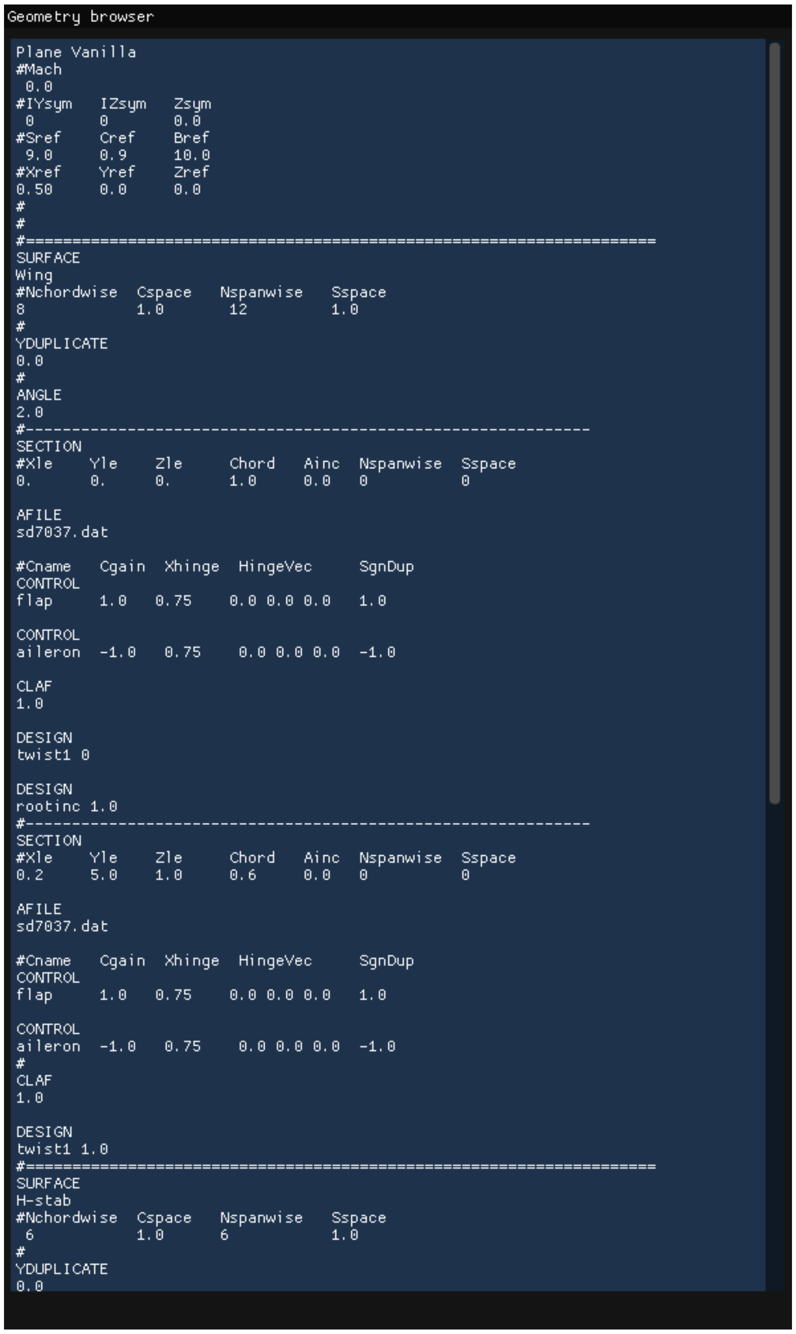


```
Plane Vanilla
#Mach
 0.0
#IYsym   IZsym   Zsym
 0       0       0.0
#Sref    Cref    Bref
 9.0     0.9     10.0
#Xref    Yref    Zref
0.50     0.0     0.0
#
#
#=====================================================================
SURFACE
Wing
#Nchordwise  Cspace   Nspanwise   Sspace
8            1.0       12         1.0
#
YDUPLICATE
0.0
#
ANGLE
2.0
#--------------------------------------------------------------
SECTION
#Xle    Yle    Zle     Chord   Ainc  Nspanwise  Sspace
0.      0.     0.      1.0     0.0   0          0

AFILE
sd7037.dat

#Cname   Cgain  Xhinge  HingeVec     SgnDup
CONTROL
flap     1.0   0.75    0.0 0.0 0.0   1.0

CONTROL
aileron  -1.0   0.75    0.0 0.0 0.0  -1.0

CLAF
1.0

DESIGN
twist1 0

DESIGN
rootinc 1.0
#--------------------------------------------------------------
SECTION
#Xle    Yle    Zle     Chord   Ainc  Nspanwise  Sspace
0.2     5.0    1.0     0.6     0.0   0          0

AFILE
sd7037.dat

#Cname   Cgain  Xhinge  HingeVec     SgnDup
CONTROL
flap     1.0   0.75    0.0 0.0 0.0   1.0

CONTROL
aileron  -1.0   0.75    0.0 0.0 0.0  -1.0
#
CLAF
1.0

DESIGN
twist1 1.0
#=====================================================================
SURFACE
H-stab
#Nchordwise  Cspace   Nspanwise   Sspace
 6           1.0      6           1.0
#
YDUPLICATE
0.0
```

These are the contents of your .avl file. Keep in mind that **the “Geometry designer” tab does not enforce .avl file conventions**; the user must take care to not break this convention while editing geometry.

The text box is editable and scrollable. More importantly, the displayed geometry updates in real time with the text box. You can make edits to your surfaces, sections, or controls. You could also add more of all three.

We’ll turn the vanilla’s wing into a gull wing by adding another section. To do this, we’ll copy and paste the wing’s second section definition.



Geometry browser

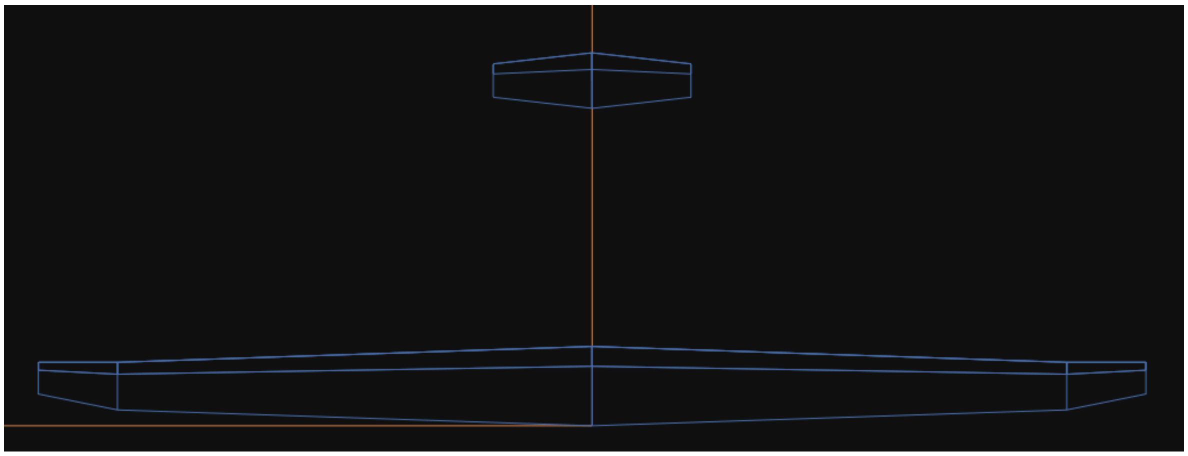
```
Plane Vanilla
#Mach
 0.0
#IYsym   IZsym   Zsym
 0       0       0.0
#Sref    Cref    Bref
 9.0     0.9     10.0
#Xref    Yref    Zref
0.50     0.0     0.0
#
#
#==================================================================
SURFACE
Wing
#Nchordwise  Cspace   Nspanwise   Sspace
8            1.0       12         1.0
#
YDUPLICATE
0.0
#
ANGLE
2.0
#-----------------------------------------------------------
SECTION
#Xle    Yle    Zle     Chord   Ainc  Nspanwise  Sspace
0.      0.     0.      1.0     0.0   0          0

AFILE
sd7037.dat

#Cname   Cgain  Xhinge  HingeVec     SgnDup
CONTROL
flap     1.0   0.75    0.0 0.0 0.0   1.0

CONTROL
aileron  -1.0   0.75    0.0 0.0 0.0  -1.0

CLAF
1.0

DESIGN
twist1 0

DESIGN
rootinc 1.0
#-----------------------------------------------------------
SECTION
#Xle    Yle    Zle     Chord   Ainc  Nspanwise  Sspace
0.2     5.0    1.0     0.6     0.0   0          0

AFILE
sd7037.dat

#Cname   Cgain  Xhinge  HingeVec     SgnDup
CONTROL
flap     1.0   0.75    0.0 0.0 0.0   1.0

CONTROL
aileron  -1.0   0.75    0.0 0.0 0.0  -1.0
#
CLAF
1.0

DESIGN
twist1 1.0
#==================================================================
SURFACE
H-stab
#Nchordwise  Cspace   Nspanwise   Sspace
 6           1.0      6           1.0
#
YDUPLICATE
0.0
```

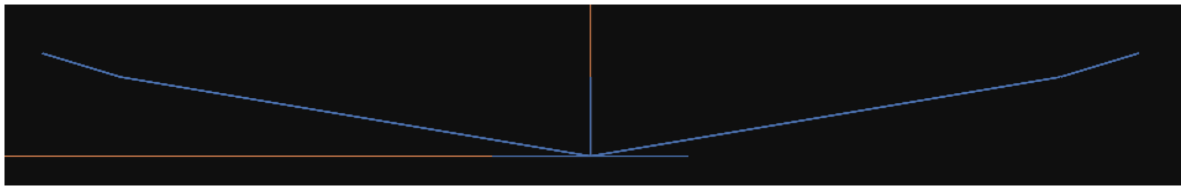
← Wing surface definition

This is the section you want

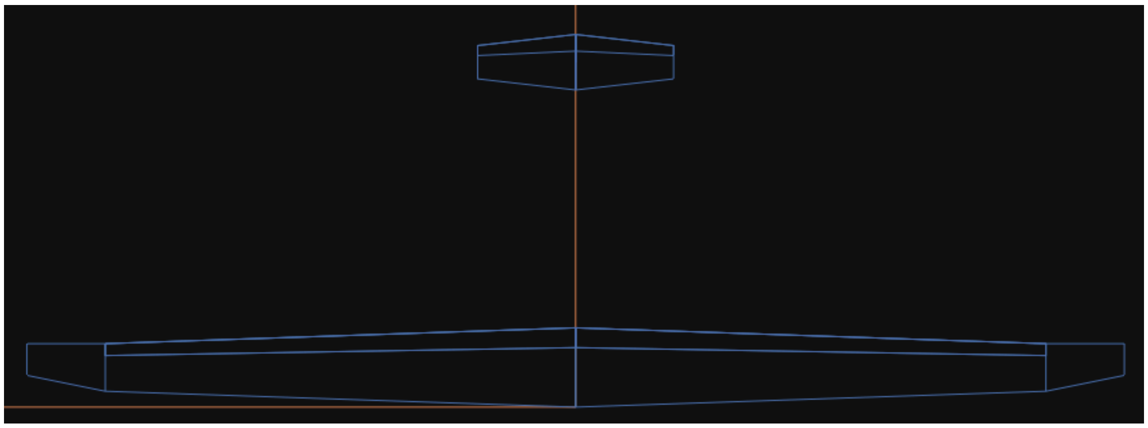
To achieve our desired gull-wing shape, we'll have to adjust the numbers in this new section. You'll have to adjust Xle, Yle, Zle, and Chord to achieve the desired shape. So here's your assignment:

Use the text field and viewing window to approximate the following planform:





Getting the exact numbers isn't terribly important. Just get something that looks similar. However, does it make sense to put ailerons/flaps on the outer ~10% of the wing? Spoiler alert: it doesn't. So let's get rid of the control definitions on the new wing section.



Now that looks like a pretty good planform. But don't lose focus too soon… I guarantee that 99.9% of users overlooked something here.

You’ve changed your aircraft planform, so **you need to update Sref, Cref, and Bref**. Did you forget? Cause I certainly did. We won’t go through the pain of recalculating all three of those, and just save the planform as is.

I suggest saving your new design to a new file, like “vanilla_gull.avl”. You could also overwrite the original vanilla.avl, but we’d like to preserve that file for the next tutorial.

To save your new design, use the “Save file” button next to the “Load file” button. This will save the text in the text field to the given filename. Saving a file will empty the text field and remove the geometry from AVL Tool.

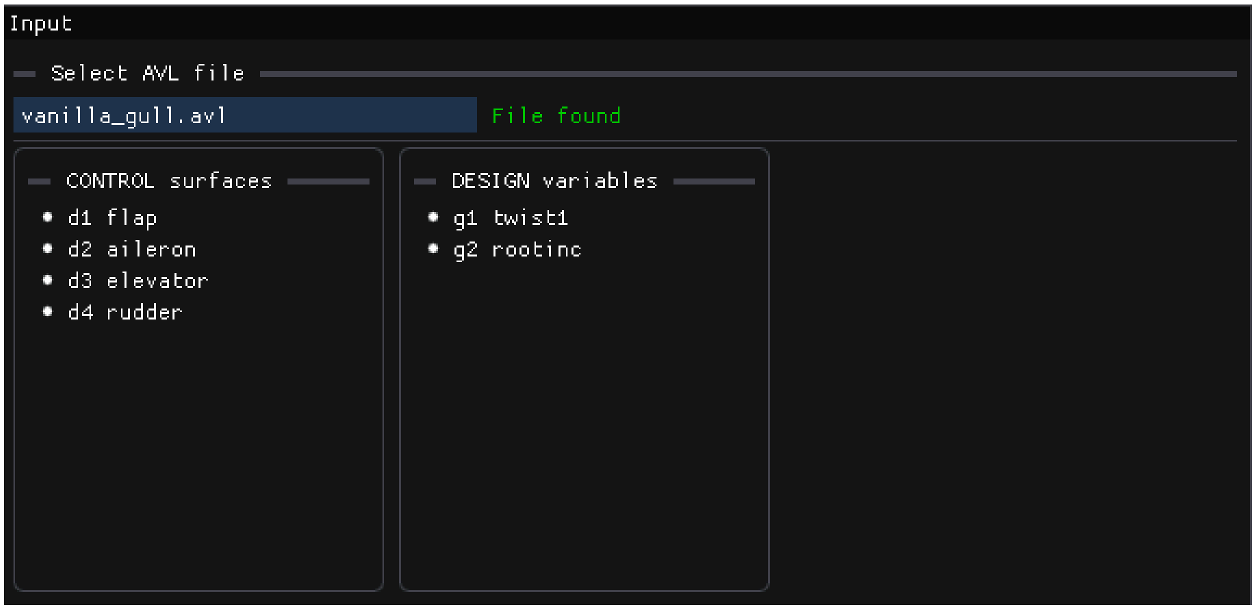
Congratulations, you’ve mastered the geometry designer.

Tutorial 2: Simulating a operating-point run case

We'll now cover how to use AVL through AVL Tool, as well as AVL Tool's utilities.

Head on over to the "AVL Tool GUI" by using the radio buttons on the right-hand side. In contrast to the geometry designer, this tab is full of buttons, text, and windows (regrettably, embedding a Subway Surfers YouTube video is beyond my capabilities).

Let's look at the top-middle window labeled "Input".



vanilla_gull.avl File found

CONTROL surfaces
- d1 flap
- d2 aileron
- d3 elevator
- d4 rudder

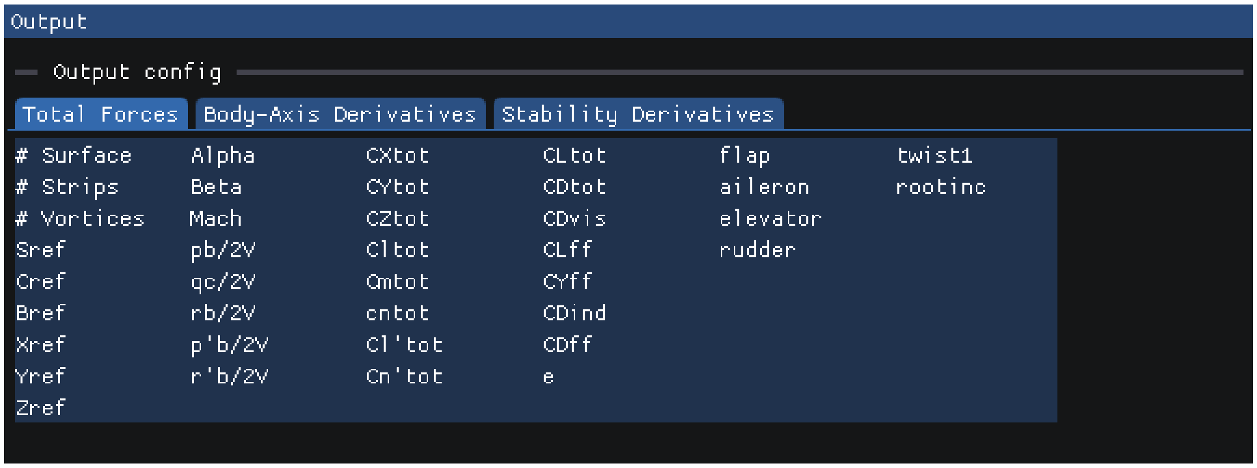
DESIGN variables
- g1 twist1
- g2 rootinc

If you just did the previous tutorial and gave your .avl file a new name, you'll notice that the name is already loaded into the text field; the two text fields are linked for the user's convenience.

For this tutorial, we'll go back to the original vanilla.avl, as its Sref, Cref, and Bref are accurate to its planform.

In this window, AVL Tool automatically displays the control surfaces and design variables found in your .avl file. This is useful for setting up a run case, as you'll know which control surface needs to trim which moment.

Moving downward to the “Output window”, we are greeted with a table of output values.


Output

Output config

Total Forces | Body-Axis Derivatives | Stability Derivatives

# Surface	Alpha	CXtot	CLtot	flap	twist1
# Strips	Beta	CYtot	CDtot	aileron	rootinc
# Vortices	Mach	CZtot	CDvis	elevator	
Sref	pb/2V	Cltot	CLff	rudder	
Cref	qc/2V	Cmtot	CYff		
Bref	rb/2V	cntot	CDind		
Xref	p'b/2V	Cl'tot	CDff		
Yref	r'b/2V	Cn'tot	e		
Zref					

The output configuration decides which of AVL’s outputs are collected for you in a file. Think of it like curbside pickup at a grocery store. Instead of going into the output files and looking for the numbers you want, have someone else do it for you (and for free! what a bargain). The output config contains three tabs, one for each of the primary output files.

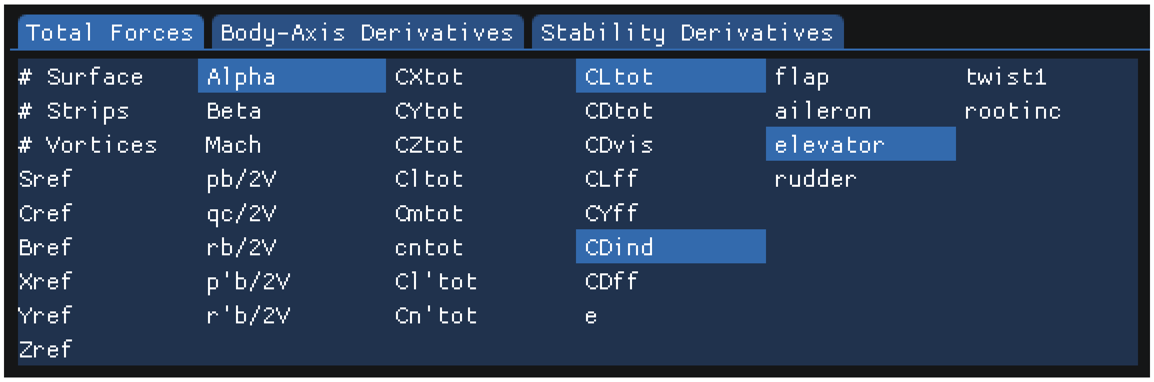
Fetching the outputs yourself isn’t a huge issue if you’re just calculating a single operating case, but when you want to sweep a domain of operating cases, automatically collecting the output for each case becomes a necessity (*ooooh foreshadowing)*.

Here are some common outputs you’ll want to keep an eye on:

- **Total Forces**: *Alpha, CLtot, CDind, elevator*
- **Stability Derivatives:** *Clb, Cma, Cnb*

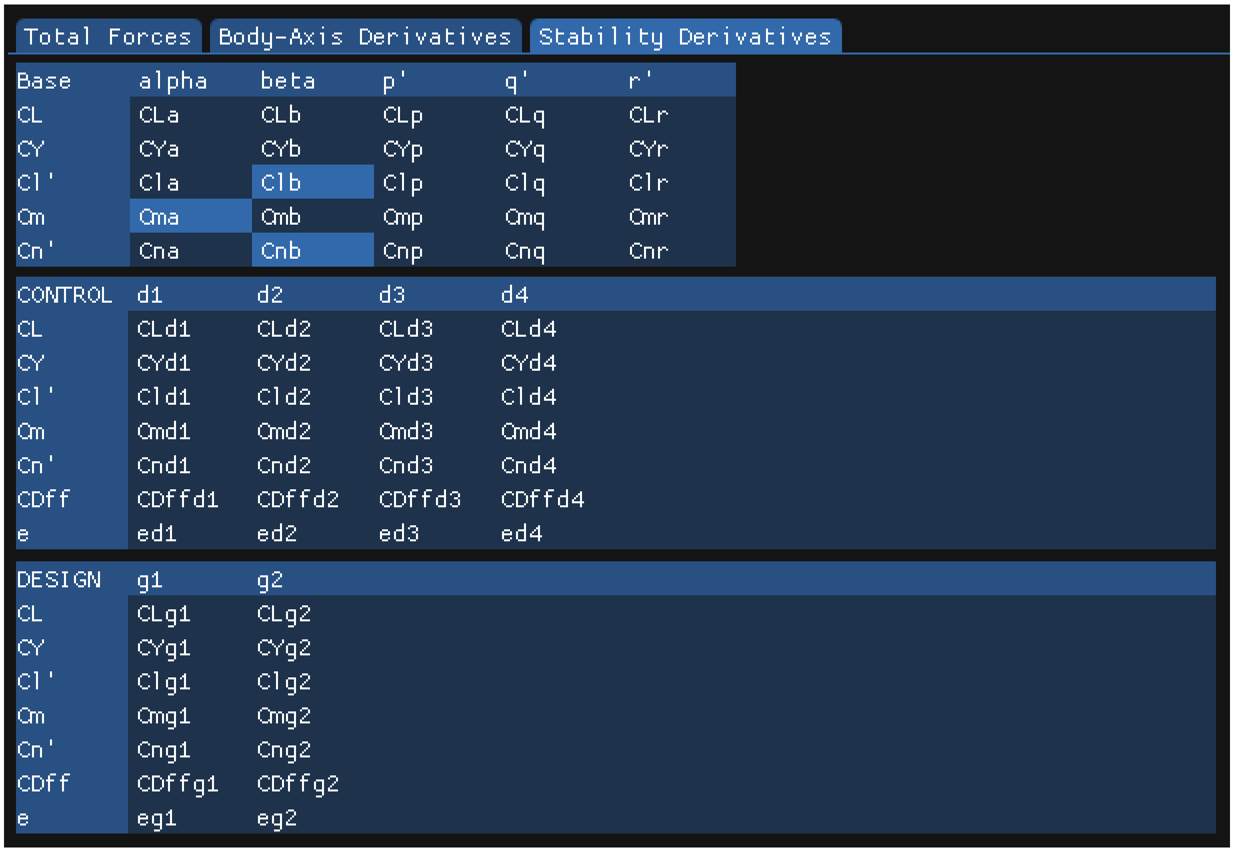
For the visual learners (me), here are those outputs:

Total Forces


Total Forces | Body-Axis Derivatives | Stability Derivatives

# Surface	Alpha	CXtot	CLtot	flap	twist1
# Strips	Beta	CYtot	CDtot	aileron	rootinc
# Vortices	Mach	CZtot	CDvis	elevator	
Sref	pb/2V	Cltot	CLff	rudder	
Cref	qc/2V	Cmtot	CYff		
Bref	rb/2V	cntot	CDind		
Xref	p'b/2V	Cl'tot	CDff		
Yref	r'b/2V	Cn'tot	e		
Zref					

Stability Derivatives



Base	alpha	beta	p'	q'	r'
CL	CLa	CLb	CLp	CLq	CLr
CY	CYa	CYb	CYp	CYq	CYr
Cl'	Cla	Clb	Clp	Clq	Clr
Cm	Cma	Cmb	Cmp	Cmq	Cmr
Cn'	Cna	Cnb	Cnp	Cnq	Cnr

CONTROL	d1	d2	d3	d4
CL	CLd1	CLd2	CLd3	CLd4
CY	CYd1	CYd2	CYd3	CYd4
Cl'	Cld1	Cld2	Cld3	Cld4
Cm	Cmd1	Cmd2	Cmd3	Cmd4
Cn'	Cnd1	Cnd2	Cnd3	Cnd4
CDff	CDffd1	CDffd2	CDffd3	CDffd4
e	ed1	ed2	ed3	ed4

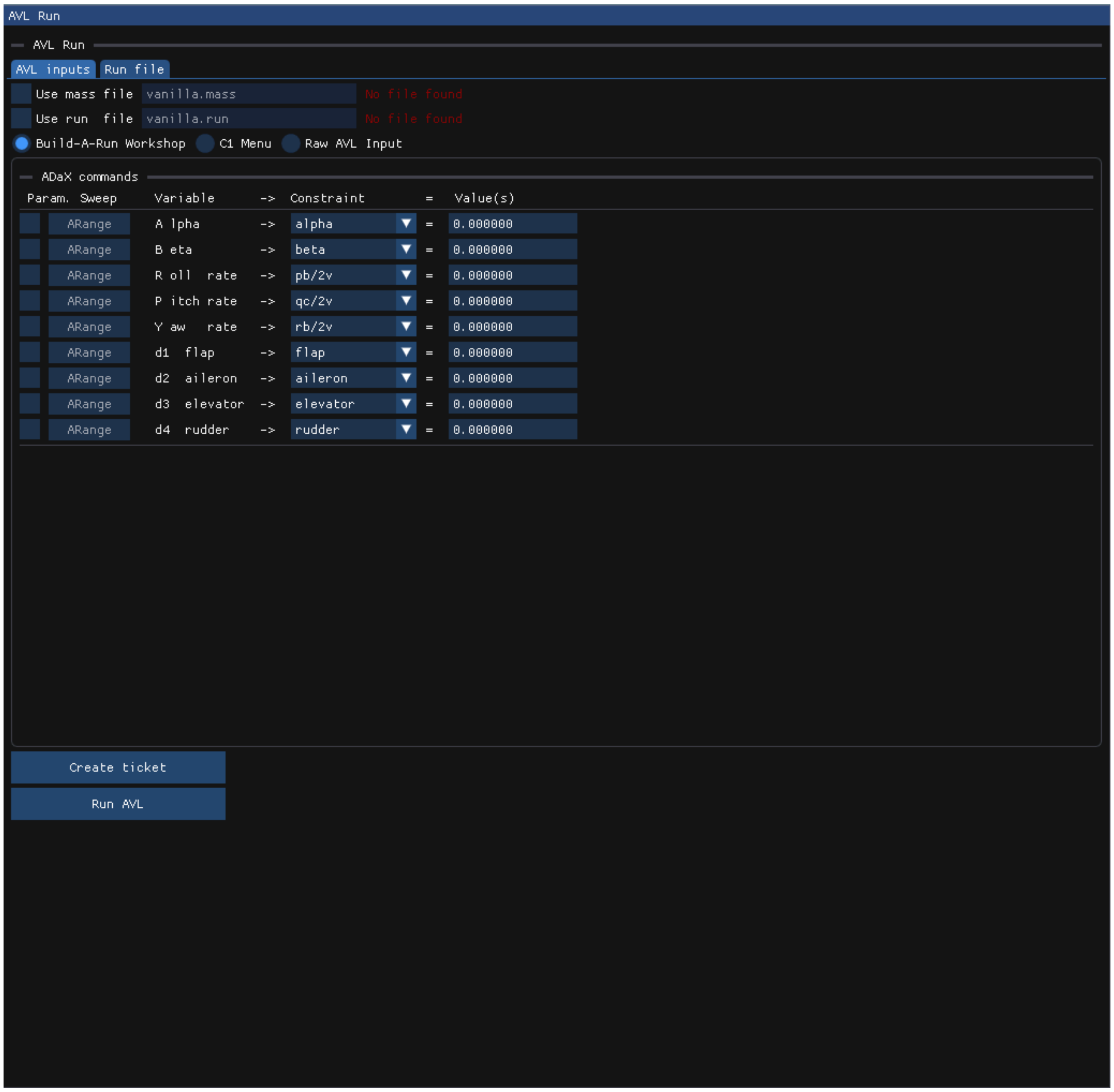
DESIGN	g1	g2
CL	CLg1	CLg2
CY	CYg1	CYg2
Cl'	Clg1	Clg2
Cm	Cmg1	Cmg2
Cn'	Cng1	Cng2
CDff	CDffg1	CDffg2
e	eg1	eg2

Everything else is on a case-by-case basis:

- Doing a convergence study? You'll want # *Strips* and # *Vortices*.
- Calculating a banked turn? Pick up *aileron* and *rudder*.
- Twist optimization? Tack on your design variables.
- Static stability on takeoff? You'll definitely want *flaps* and *slats*.
- Stall calculation? AVL can do no such thing.

For now, we'll stick with a straight, level, unaccelerated flight (SLUF) calculation. Set up the output config in the Output window using the seven aforementioned values and we'll get to setting up that run case.

Let's take a look at the biggest window, the "AVL Run" window. This window handles everything related to "what goes into AVL's command line". There are two tabs: the "AVL inputs" and "Run file".


AVL Run

AVL Run

AVL inputs | Run file

Use mass file vanilla.mass No file found

Use run file vanilla.run No file found

[✓ Build-A-Run Workshop] C1 Menu Raw AVL Input

ADaX commands

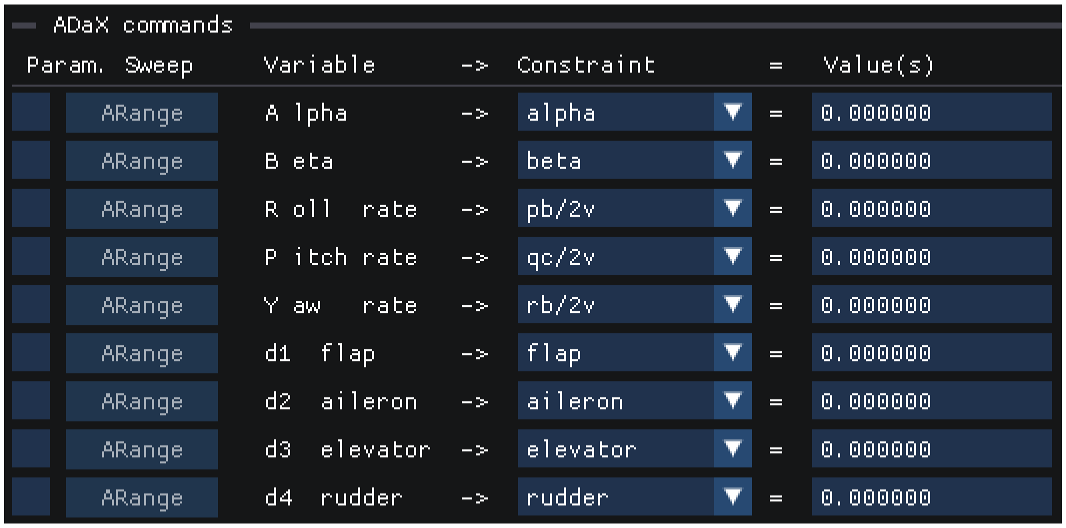
Param.	Sweep	Variable	->	Constraint	=	Value(s)
	ARange	A lpha	->	alpha	=	0.000000
	ARange	B eta	->	beta	=	0.000000
	ARange	R oll rate	->	pb/2v	=	0.000000
	ARange	P itch rate	->	qc/2v	=	0.000000
	ARange	Y aw rate	->	rb/2v	=	0.000000
	ARange	d1 flap	->	flap	=	0.000000
	ARange	d2 aileron	->	aileron	=	0.000000
	ARange	d3 elevator	->	elevator	=	0.000000
	ARange	d4 rudder	->	rudder	=	0.000000

We’ll be working with “AVL inputs” for now, while the “Run file” tab is a WIP, disabled, very-likely-obsolete, soon-to-be-removed feature that caused me nothing but headaches. But it might find a use. Anyone’s guess, really.

At the top of the AVL inputs tab, we have options to use mass and run files. We won't need them for our purposes here. Below that, we have radio buttons to choose between three tools:

- **Build-A-Run (BAR) Workshop:** for creating operating-point run cases or sweeping a range of run cases.
- **C1 Menu:** Tool for calculating the trim CL and bank angular rates. This is an upgraded version of AVL's C1 submenu.
- **Raw AVL Input:** For users who want to manually create a reusable input file for AVL.

For calculating SLUF, we'll make use of the BAR and C1 tabs. Let's familiarize ourselves with the BAR first.



ADaX commands

Param.	Sweep	Variable	->	Constraint	=	Value(s)
	ARange	A lpha	->	alpha	=	0.000000
	ARange	B eta	->	beta	=	0.000000
	ARange	R oll rate	->	pb/2v	=	0.000000
	ARange	P itch rate	->	qc/2v	=	0.000000
	ARange	Y aw rate	->	rb/2v	=	0.000000
	ARange	d1 flap	->	flap	=	0.000000
	ARange	d2 aileron	->	aileron	=	0.000000
	ARange	d3 elevator	->	elevator	=	0.000000
	ARange	d4 rudder	->	rudder	=	0.000000

The first column contains a checkbox and button. The checkbox enables parameter sweeping, with the button changing how the domain is defined. We'll use this in tutorial 3. **For now, leave all the boxes unchecked**.

The second column contains the variables. **There are five independent variables that make up your aircraft's (unaccelerated) state: *alpha, beta, roll rate (p), pitch rate (q),* and *yaw rate (r)*.** Your control surface deflections are also independent variables, controlled by the pilot. For our vanilla plane, we have four control surfaces.

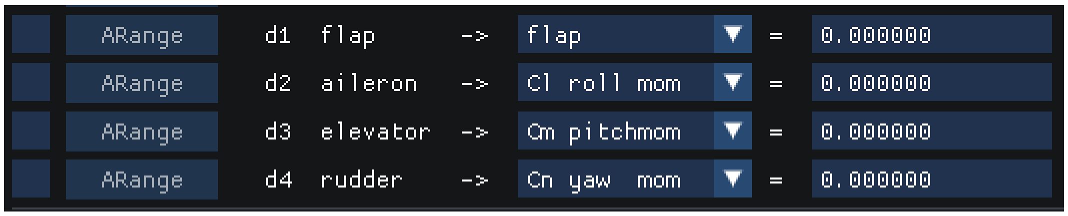
The third column are the constraints. Each independent variable can be set to control one of the ten default (plus # of control surfaces) constraints. Each variable has an equivalent constraint. By setting the variable equal to its corresponding constraint, you are simply driving that variable to be some value. To elaborate:

"`Alpha → alpha = 0.00`" is saying "set alpha to 0.0".
On the other hand,
"`Alpha → CL    = 0.43`" is saying "change alpha *such that* CL is 0.43".

By default, all variables are set to zero in the oper menu. It is up to you to change the constraints and values to achieve your desired state.

The fourth column is the value. This would be the 0.0 and 0.43 in the previous examples.

For SLUF, we need to trim our aircraft. This means that the moment coefficients are zero and the lift coefficient balances the weight coefficient (i.e., *L* = *W*). To trim the moments, use the control surfaces to set the corresponding moments to zero.

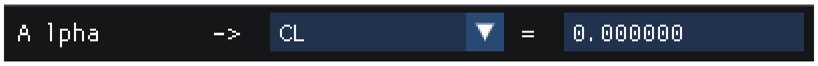


```
ARange   d1  flap      ->  flap         =  0.000000
ARange   d2  aileron   ->  Cl roll mom  =  0.000000
ARange   d3  elevator  ->  Cm pitchmom  =  0.000000
ARange   d4  rudder    ->  Cn yaw  mom  =  0.000000
```

The flaps are set to zero, as SLUF is a cruise configuration, where we are unlikely to be deploying our flaps. The vanilla plane actually has "*flaperons*", where the flaps and ailerons are the same control surface. Since we are not doing a takeoff, landing, or aerial delivery, we won't stress the details here.

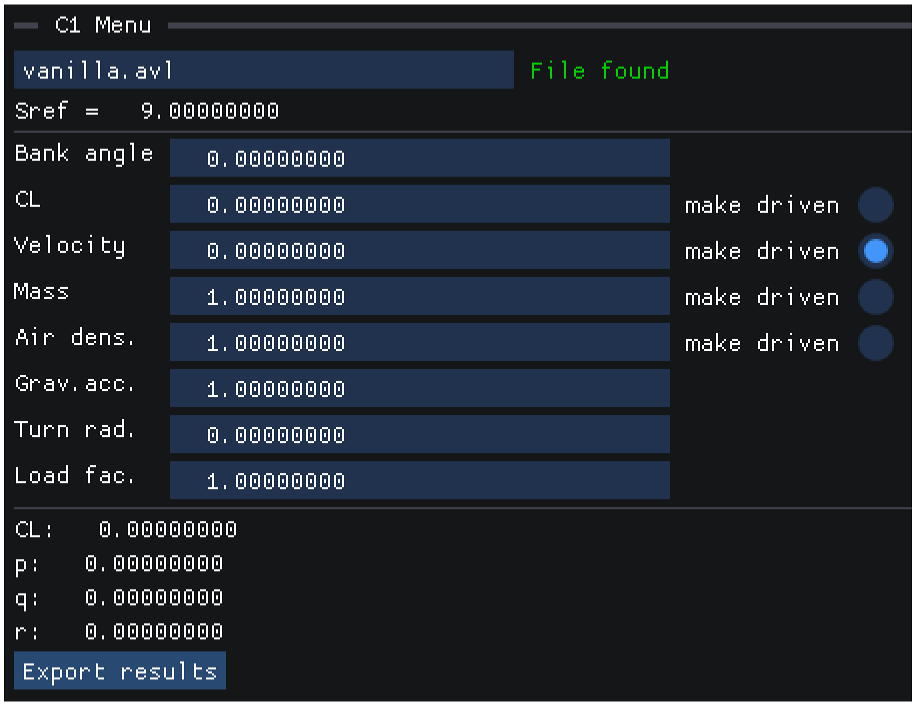
The aileron drives the rolling moment to zero, the elevator drives the pitching moment to zero, and the rudder drives the yawning moment to zero. Without a sideslip angle, the aileron and rudder will not be deflected unless 1) your plane is asymmetric or 2) your geometry is messed up. However, it's good practice to remember that you need to trim all three moments.

To trim our forces (*L* = *W*), we'll need to change our angle of attack to achieve some cruise CL.



```
A lpha        ->  CL ▼  =  0.000000
```

The only problem is that we don't know the cruise CL… and it sure isn't zero. So let's head over to the C1 tab.



```
C1 Menu
vanilla.avl                                File found
Sref =   9.00000000
Bank angle   0.00000000
CL           0.00000000    make driven ○
Velocity     0.00000000    make driven ●
Mass         1.00000000    make driven ○
Air dens.    1.00000000    make driven ○
Grav.acc.    1.00000000
Turn rad.    0.00000000
Load fac.    1.00000000
CL:   0.00000000
p:   0.00000000
q:   0.00000000
r:   0.00000000
Export results
```

At the top we have our .avl file. We need to know this because the required CL depends on Sref. This is why we chose to go with the original vanilla.avl. In the C1 menu, we have a variety of variables related to our flight conditions. The bank angle is only important for banked turns, of which SLUF is not. Defining your bank angle determines your turn radius and load factor. The remaining variables are related to finding our CL.

The "make driven" radio buttons determine which variable is subject to change when the others do. For AVL, velocity is always changed when the user changes any other parameter (when the user changes velocity, CL changes to compensate).

However, this is bothersome if mission requirements have a required airspeed. Therefore, AVL Tool's C1 menu allows the user to change which variables are held constant. Here are some examples of choosing your driven variable:

- **CL is driven**: must fly at a certain altitude and speed; payload mass determines CL.
- **Velocity is driven**: designers choose airspeed and would like to maintain best CL
- **Air density is driven**: velocity required and want to maintain best CL, therefore density (cruise altitude) changes.
- **Mass is driven**: Find what mass you need to be at for best performance (best CL) at a given altitude and speed.

The examples are not exhaustive, so choose the best driven variable for your requirements. Do take note of how "best CL" appears in every example except ***CL is driven***.

Enough small talk. Let's set up our flight conditions. For this tutorial, we'll work in imperial units (*collective booing from the audience)* because they are more confusing than SI; this will serve as an example of how to handle imperial units in AVL. (*semi-collective nodding from the audience*). Here are our LMT units:

Length:	feet
Mass:	slug
Time:	second

This means that **our plane's mass will be its weight in pounds divided by 32.17**.

We don't know what our best CL is yet. That would require getting a drag polar and finding the maximum CL. It's almost like you'd need to sweep through a range of alpha… using AVL… (*oooooh shameless foreshadowing*).

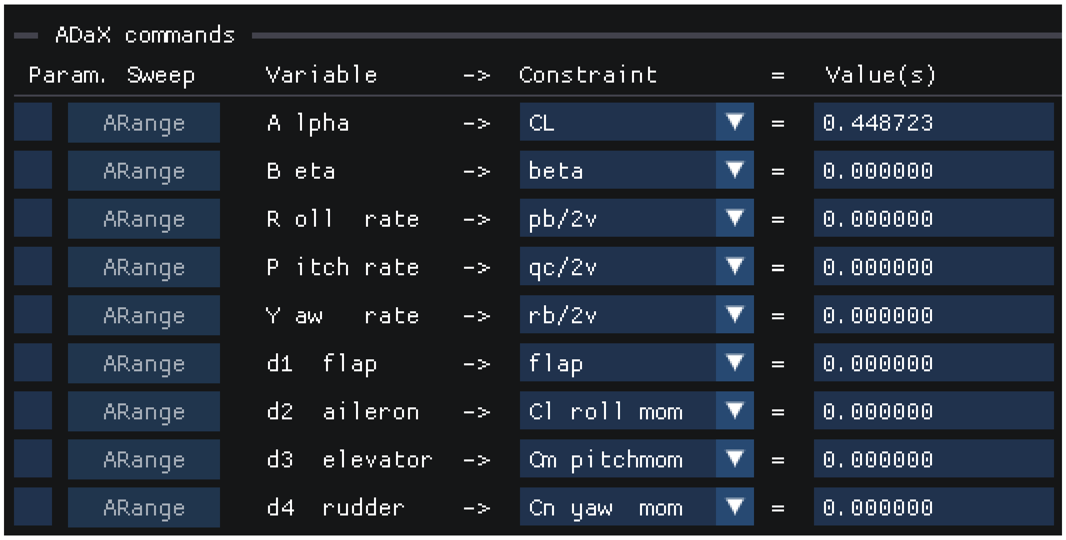
So for now, we'll make CL the driven parameter. Press the "make driven" radio button next to the CL entry.

For a 10-foot span plane in a RC-plane-like configuration, it wouldn't be so farfetched to say that the plane weighs ~12 pounds and flies near sea level. Throw in 32.17 for gravity (ft/s^2), 23.77e-4 for density (slug/ft^3), and 0.373 for mass (slug).

If you do it right, you'll get the very-achievable CL of `inf`. Of course it's infinity, we're trying to fly at zero feet per second.

A good cruise CL will be around 0.3-0.5. Setting our velocity to 50 ft/s will yield a CL of around 0.45. That's pretty reasonable. Feel free to play with the velocity a bit, or spice it up by making other parameters driven.

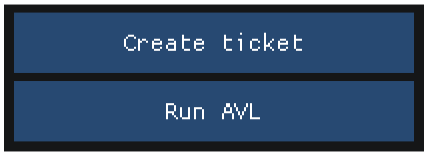
Armed with our new CL, press the "Export values to BAR". Return to the BAR, and you should now have your angle of attack driving your CL.



ADaX commands

Param.	Sweep	Variable	->	Constraint	=	Value(s)
	ARange	A lpha	->	CL	=	0.448723
	ARange	B eta	->	beta	=	0.000000
	ARange	R oll rate	->	pb/2v	=	0.000000
	ARange	P itch rate	->	qc/2v	=	0.000000
	ARange	Y aw rate	->	rb/2v	=	0.000000
	ARange	d1 flap	->	flap	=	0.000000
	ARange	d2 aileron	->	Cl roll mom	=	0.000000
	ARange	d3 elevator	->	Cm pitchmom	=	0.000000
	ARange	d4 rudder	->	Cn yaw mom	=	0.000000

And with that, our plane is fully trimmed for SLUF. You can now turn your attention to the strangely-placed and oddly-large "Create ticket" and "Run AVL" buttons below the BAR.



Please bear with me while I find a better place for these buttons.

Once you've created a run case (or domain sweep) in the BAR, you'll need to create a ticket. The ticket includes the instructions telling AVL Tool how to create the set of AVL instructions and which outputs to collect. Think of it like an order at a restaurant (or perhaps… a BAR? (*more collective booing ensues. A well placed tomato hits the would-not-be comedian in the forehead*).

You (user) place an order such as “burger, small fries, chocolate shake” (create a ticket). This information (set of commands in input_list.txt) is relayed to the chef (avl.exe), who cooks up a burger, scoops your fries, and whips up a chocolate shake (calculations and output files). This is all assembled on the tray (OUTPUT.dat file) which is handed over to you (still the user).

A ticket only needs to be created once, then you can run that same set of instructions in AVL as many times as you want.

Press the “Create ticket” button. If all goes well, the command line for the GUI will read:

```
Creating ticket file... done
```

If something else happens, please let me know so I can continue my error-handling conquest.

The ticket file is written to “WrapperTicket.ticket”. It is a plaintext file. You could probably read it and get the gist of what’s going on.

With your ticket file made, you can now press the “Run AVL” button to parse and run your ticket file. The GUI command line will spit out some nonsense and run AVL for you. If it encounters an error, it will spit out something like:

```
Error code: (number here)
```

Again, if you get an error code, please let me know so I can full nelson the error and suplex its dreams of ever occurring again.

Now check out the OUTPUT.dat file. It contains the outputs you specified in the output configuration, using the numbers obtained from your BAR-created run case.

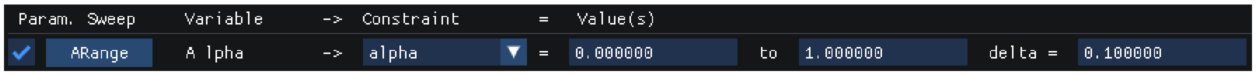
Tutorial 3: Variable sweeps and inviscid polars

It's time to up the ante. Unlike XFOIL, AVL does not have an inherent option to sweep through a sequence of alphas and spit out a polar. This is mildly inconveniencing. Not a single person has ever woken up and thought "*nah I don't need a drag polar*". If they have, they are sorely misled.

So let's just jump right into using AVL Tool to create a drag polar. Close and reopen AVL Tool just so you have a fresh, silvery landscape to work with.

We'll keep working with vanilla.avl. For our output, we'll want *alpha, CLtot, CDind,* and *elevator*.
Next, head over to the BAR and trim the moments. Doing this again from scratch will fast track familiarizing yourself with AVL Tool.

After that, we'll make use of the parameter sweep feature. Check the "Param. Sweep" box for alpha. Leave alpha as driving itself. (i.e., NOT "Alpha → CL")



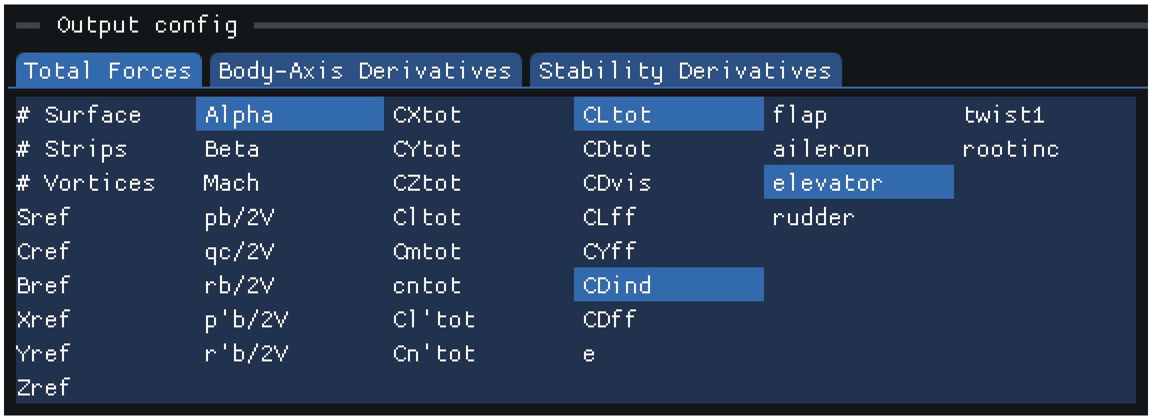
```
Param. Sweep   Variable   ->  Constraint   =  Value(s)
✓ ARange       Alpha      ->  alpha ▼      =  0.000000   to  1.000000   delta =  0.100000
```

Checking the box will expand your value options to include a start value, end value, and gridding information. You have two options for defining the domain: LinSpace (define a number of points) and ARange (define a delta). To switch between the two, click the button next to the parameter sweep checkbox.

We'll stick with ARange for this example. Choose your lower and upper bounds for your inviscid polar. A safe bet would be from -15 to 15 degrees. As for the delta, this is up for you and your team to decide. If you want your inviscid polar to work nicely with your parasite drag polar, you'll want to use the same delta/Npoints.

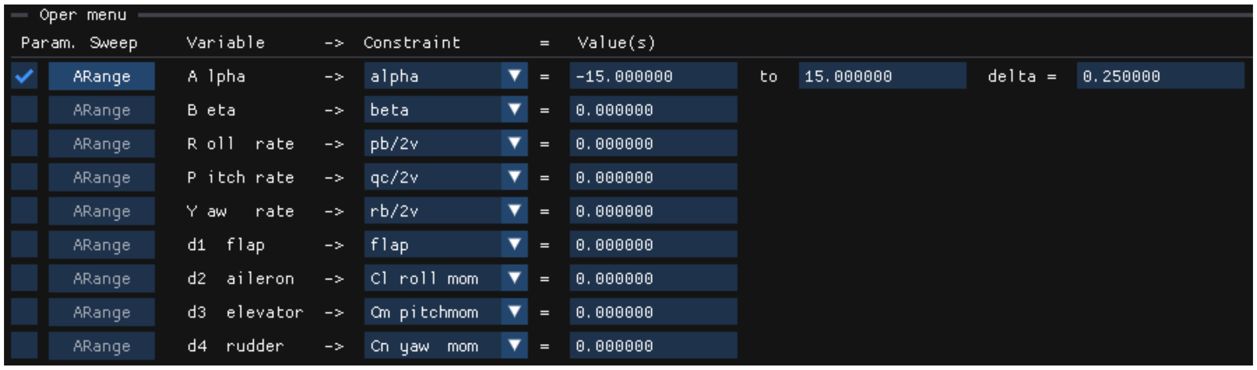
Since the disembodied voice walking you through this tutorial has no such parasitic drag polar shackling their unmatched and unrelenting creative brilliance, we'll go with a delta of 0.25.

When you're done, your output config and BAR should look like this:


Output config

Total Forces | Body-Axis Derivatives | Stability Derivatives

# Surface	[✓ Alpha]	CXtot	[✓ CLtot]	flap	twist1
# Strips	Beta	CYtot	CDtot	aileron	rootinc
# Vortices	Mach	CZtot	CDvis	[✓ elevator]	
Sref	pb/2V	Cltot	CLff	rudder	
Cref	qc/2V	Cmtot	CYff		
Bref	rb/2V	cntot	[✓ CDind]		
Xref	p'b/2V	Cl'tot	CDff		
Yref	r'b/2V	Cn'tot	e		
Zref					


Oper menu

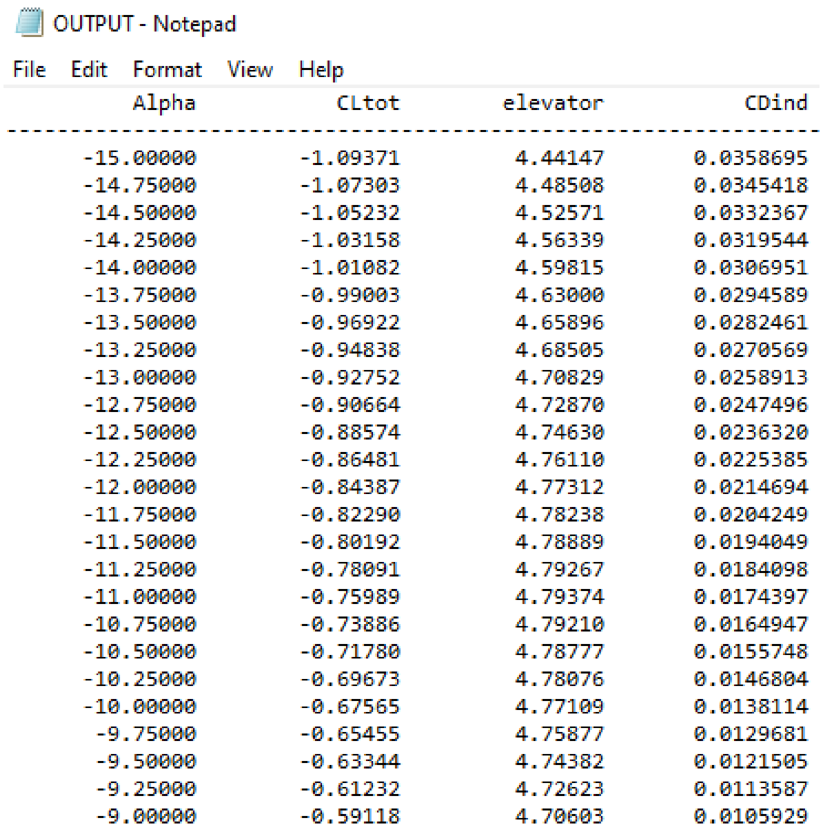
Param.	Sweep	Variable	->	Constraint	=	Value(s)			
✓	ARange	A lpha	->	alpha	=	-15.000000	to	15.000000	delta = 0.250000
	ARange	B eta	->	beta	=	0.000000			
	ARange	R oll rate	->	pb/2v	=	0.000000			
	ARange	P itch rate	->	qc/2v	=	0.000000			
	ARange	Y aw rate	->	rb/2v	=	0.000000			
	ARange	d1 flap	->	flap	=	0.000000			
	ARange	d2 aileron	->	Cl roll mom	=	0.000000			
	ARange	d3 elevator	->	Cm pitchmom	=	0.000000			
	ARange	d4 rudder	->	Cn yaw mom	=	0.000000			

We know that you don't need to trim all three moments, but it's still good practice to do so.

Create that ticket and check the command line to ensure it was generated properly. Check out the ticket file if you'd like. Again, you can look at it and get the gist of what it wants to do.

With the ticket file created, run AVL. The application will likely freeze up and fill your command line with nonsense as it sweeps through the specified domain. The more points in your grid, the longer it'll take. I do have ideas to prevent the freezing while doing a parameter sweep, but that's for future versions.

Check out the OUTPUT.dat file. It now contains your inviscid polar information.



```
         Alpha          CLtot        elevator         CDind
-----------------------------------------------------------------
     -15.00000       -1.09371         4.44147     0.0358695
     -14.75000       -1.07303         4.48508     0.0345418
     -14.50000       -1.05232         4.52571     0.0332367
     -14.25000       -1.03158         4.56339     0.0319544
     -14.00000       -1.01082         4.59815     0.0306951
     -13.75000       -0.99003         4.63000     0.0294589
     -13.50000       -0.96922         4.65896     0.0282461
     -13.25000       -0.94838         4.68505     0.0270569
     -13.00000       -0.92752         4.70829     0.0258913
     -12.75000       -0.90664         4.72870     0.0247496
     -12.50000       -0.88574         4.74630     0.0236320
     -12.25000       -0.86481         4.76110     0.0225385
     -12.00000       -0.84387         4.77312     0.0214694
     -11.75000       -0.82290         4.78238     0.0204249
     -11.50000       -0.80192         4.78889     0.0194049
     -11.25000       -0.78091         4.79267     0.0184098
     -11.00000       -0.75989         4.79374     0.0174397
     -10.75000       -0.73886         4.79210     0.0164947
     -10.50000       -0.71780         4.78777     0.0155748
     -10.25000       -0.69673         4.78076     0.0146804
     -10.00000       -0.67565         4.77109     0.0138114
      -9.75000       -0.65455         4.75877     0.0129681
      -9.50000       -0.63344         4.74382     0.0121505
      -9.25000       -0.61232         4.72623     0.0113587
      -9.00000       -0.59118         4.70603     0.0105929
```

The order that the parameters appear in may not be ideal, but it's unlikely that I'll change how they're ordered. You can now take this file and import it into your Python script. Enjoy your new drag polar!