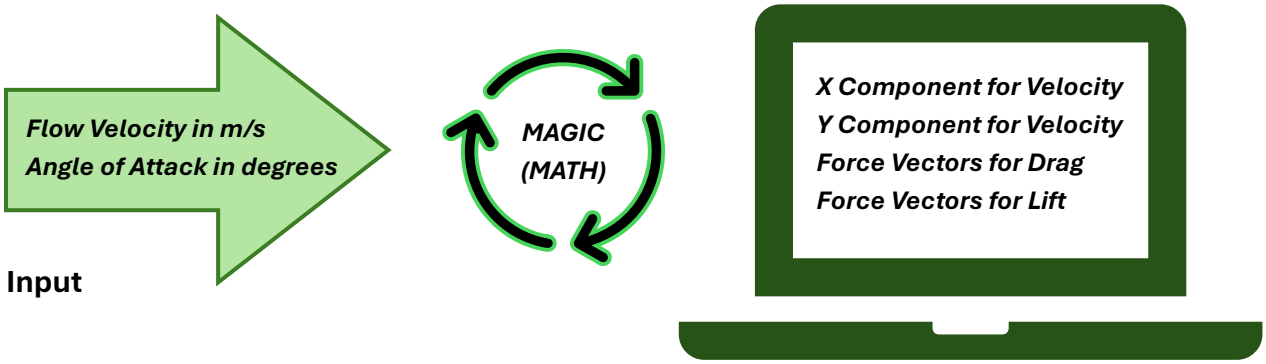
Python Script for ANSYS Settings to Manipulate Flow Direction (Angle of Attack)

Description: A simple python script (calculator) that returns some key values to be used for manipulating flow direction (AoA) in the settings of ANSYS Fluent simulation.

Keywords: *CFD, AoA, ANSYS, Python*

Overview of the Code



Input

Context

In ANSYS fluent, you can manipulate angle of attack (i.e., the angle between flow stream and the chord line of an airfoil) for simulations to calculate lift, drag, Cp, WSS and other relevant fluid mechanical properties for a drag inducing (or lift generating) object. There are two ways to do this. The first one is obvious, rotate the object with respect to the X-axis of the cross-section plane of the fluid domain. But in order to do that, you have to start before setting up the mesh, which not only takes more time, but also computationally inefficient. There is another, much faster way to do this, and that is by using vectors to alter the direction of flow. If you change a few settings in the boundary conditions and report definitions (i.e., force vectors for lift and drag), you can run the simulation in a way that will feel like the flow is hitting the object from an angle. In other words, Angle of Attack will be considered for the solver.

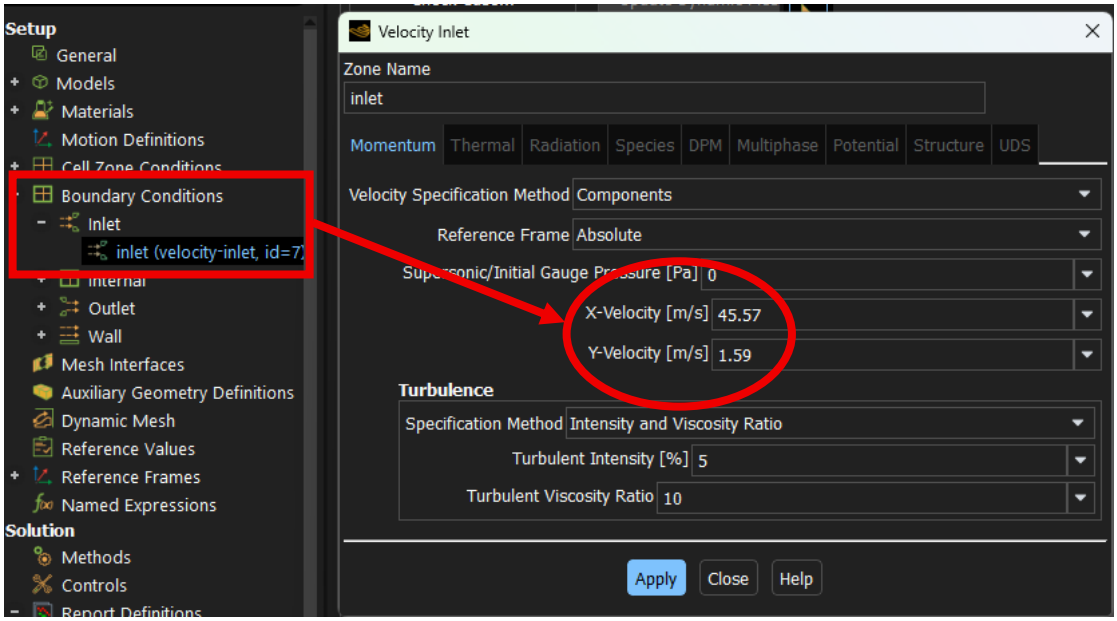
If any of this is confusing at this point, please refer to this section where the settings of a computational fluid dynamics (for a 2D airfoil) have been discussed.

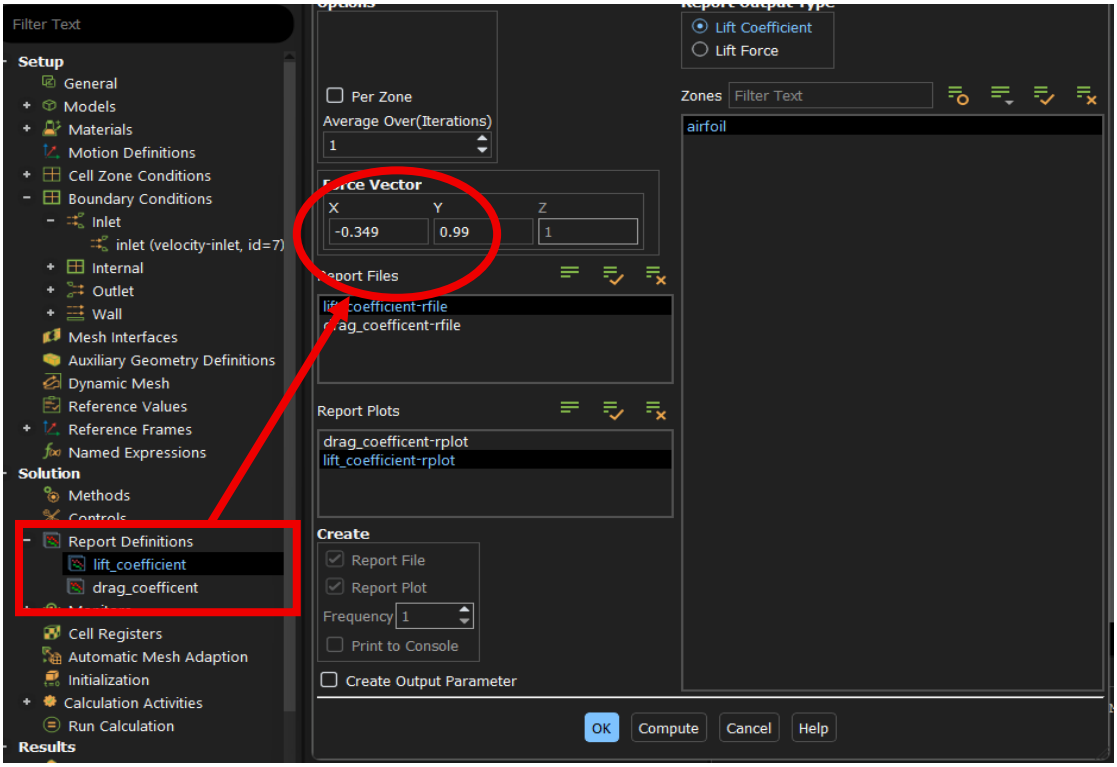
Now, our goal is to change the settings in a way so that the flow mimics our desired Angle of Attack in the solver.

Here are the settings that require our attention –

- Boundary Conditions – Inlet – Velocity Specification Method – Components
  - X-Velocity:
  - Y-Velocity:
- Report Definitions – Lift and Drag Coefficient (assuming we have set these already) – Force Vector

There are 6 settings that require to be changed in order to mimic a certain Angle of Attack. Please see the images below for a clearer understanding –



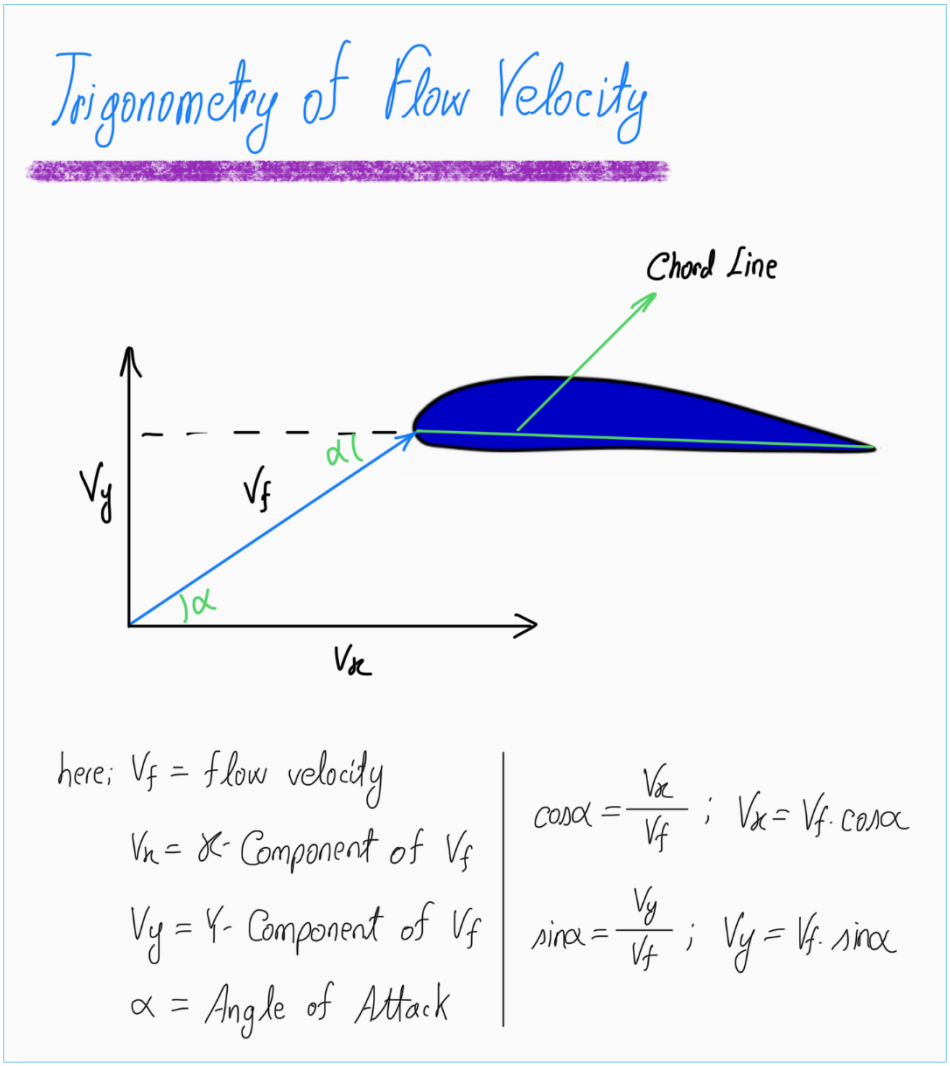


If we know our fluid velocity and the AoA we want to mimic, we can just make the adjustments as such and 'rotate' the object without rotating.

But what does Python have to do with any of this?

I am glad you asked (or I assume you asked!).

Now, in order to know the values of these settings, we rely on a neat little mathematical trick called vectors. With some trigonometric function, you can 'split' a velocity in a way that replicates an 'angled' flow stream exacerbating the desired Angle of Attack. Since velocity is a vector, we can modify it's magnitude in two different direction (X and Y) on the plane and represent the angle with the resultant velocity vector. Here is a quick refresher of what I mean by that (excuse my crude penmanship) –



Now, as you can see, I need to do some calculations every time I want to change the Angle of Attack and trust me on this – every time I want to make those calculations, I cannot find

my graphene calculator! It was frustrating at first, and annoying in the long run. I don't understand why the calculator dips out of existence every time I need it? In other times, it just sits randomly in my vicinity. Casio did not make that calculator, Houdini did.

That is the whole reason (true story) I decided to 'semi-automate' this whole process using a python script. In this script, the program will ask you to input the freestream velocity and your desired Angle of Attack. And through magic (math), it will return all the 6 values you need to change in the settings in order to mimic that flow in that particular AoA.

Python Script: The Last AoA Bender

If you read this so far, congratulations. You just spent 5 minutes reading some random stuff before getting into the point. If not, good. You are an efficient individual who wants to get to the point. Either way, let's dig in –

```python
import math

ad = float(input("Angnle of Attack in Degrees: "))
ar = math.radians(ad)
sv = math.sin(ar)
cv = math.cos(ar)
v = float(input("Velocity in m/s: "))

Vx = v*cv                   #Velocity Component along X-axis
Vy = v*sv                   #Velocity Component along Y-axis

Ls = -1*sv                  #Negative operator for Lift Vector

print(f"X Component for Velocity = {Vx:.4f}")
print(f"Y Component for Velocity = {Vy:.4f}")
print("\nForce Vector for Drag: ")
print(f"X = {cv:.4f}     Y = {sv:.4f}")
print("\nForce Vector for Lift: ")
print(f"X = {Ls:.4f}     Y = {cv:.4f}")
```

```
Angnle of Attack in Degrees: 21
Velocity in m/s: 58
X Component for Velocity = 54.1477
Y Component for Velocity = 20.7853

Force Vector for Drag:
X = 0.9336     Y = 0.3584

Force Vector for Lift:
X = -0.3584     Y = 0.9336
```

The code will ask for velocity (in m/s) and Angle of Attack (in degrees). With your input, the code will do some basic operations (degree to radian, trigonometric functions) and spit out the settings value in a streamlined and easy to understand way. You just copy paste the values in the appropriate sections and you are done.

Welcome to the world of semi-automated Angle of Attack manipulation (is that even a thing?).

If you'd like to see this in action or build upon it, the code will be available on my GitHub. I will update this manual with a link to that soon.

Thank you for your attention to this matter. Let's make some airfoils sing (do airfoils sing or hum?).

What is Next

Although a simple Python script, the underlying logic can be used in an experimental setup to modify AoA for specimen in a wind tunnel with a microcontroller and some gimble movement (an actuator motor, for example). Considering a setup where we extract simple mathematical models for aerodynamic performances through data-driven methods, the same 'not so high-tech python script' can offer a gateway to researchers in building the particular component for experimental use cases.

Omar Saif
CFD Enthusiast
August 2, 2025