EAE 127 Applied Aircraft Aerodynamics

Project 0
Introduction to Aerodynamics and Python

DUE: Monday 10/19/2020 2:00pm

Fig. 1: Cessna 172 general aviation aircraft

This assignment will function as a general introduction to some fundamental aerodynamic concepts as well as an introductory exercise in learning Python. Perform the calculations for the following **bolded** tasks in a Python script and present your results in report format. **(NOTE: The solution has been provided beforehand so you have an example to learn from)**

1 Boundary Layers and Numeric Integration

1.1 Turbulent Boundary Layer Velocity Profile

For the first portion of this problem, we will calculate and plot the horizontal velocity distribution in the vertical direction of a turbulent boundary layer. The purpose of this exercise is to illustrate concepts pertaining to boundary layers while developing python skills in creating and manipulating data.

The *non-dimensional* velocity profile within a turbulent boundary layer can be approximated with the following relation:

$$\frac{u}{u_e} \approx \left(\frac{y}{\delta}\right)^{\frac{1}{7}}$$

Plot the boundary layer velocity profile, with $\frac{u}{u_e}$ on the horizontal axis and $\frac{y}{\delta}$ on the vertical axis.

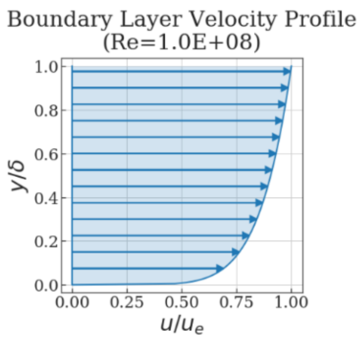

Fig. 2: Example of non-dimensional velocity profile of a turbulent boundary layer

Using the Numpy function "linspace", create the data for this plot, make a non-dimensional height variable $ynon = \frac{y}{\delta}$ that ranges from 0 (the wall) to 1 (the edge of the boundary layer). Plug this non-dimensional variable into the above equation to calculate $unon = \frac{u}{u_e}$, and then plot *unon* vs *ynon* (See Fig. 2).

1.2 Boundary Layer Thickness

Now that we have a concept of the flow behavior inside of the boundary layer, we can use this information to determine other global properties about the boundary layer, such as the thickness of the boundary layer δ, which is the height of the edge of the boundary layer above the surface as some location x or the displacement thickness δ^* which is the distance a potential streamline is displaced by the boundary layer at some location x.

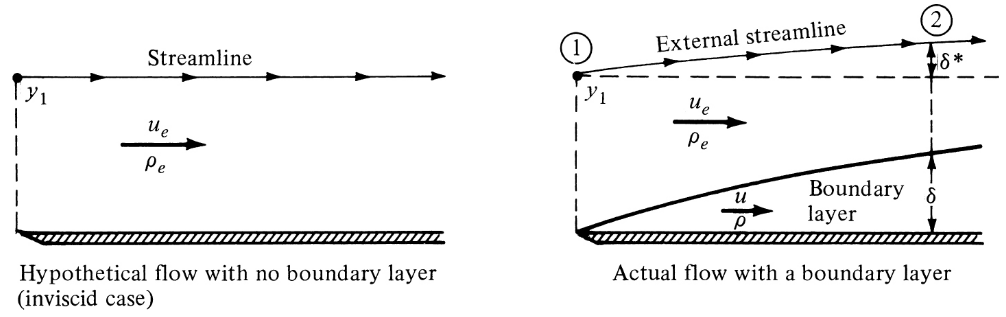

Fig. 3: Diagram (right) of boundary layer height δ and displacement thickness δ^* at a specific x-location along a flat plate (*Source: Anderson's Fundamentals of Aerodynamics, Figure 17.5*)

First, **estimate and report the boundary layer thickness** δ at the aft end of a Boeing 747 ($x = 300ft$, $Re_x = 1e8$) with the following equation:

$$\delta(x) = \frac{0.16x}{(Re_x)^{\frac{1}{7}}}$$

where Re_x is the Reynolds number at the location x. Remember that exponents in Python use the '$**$' operator (as opposed to '^').

Next, **calculate the displacement thickness** δ^* at this location by <u>numerically</u> integrating the following equation using Numpy's "trapz" function:

$$\frac{\delta^*}{\delta} = \int_0^1 \left(1 - \frac{u}{u_e}\right) d\frac{y}{\delta}$$

where $\frac{y}{\delta} = ynon$ and $\frac{u}{u_e} = unon$ are the same variables from the previous problem. You will need to redimensionalize δ^* by multiplying the result of the above equation by δ:

$$\delta^* = \frac{\delta^*}{\delta} \cdot \delta$$

2 Airfoil Plotting and Line Integrals

2.1 Airfoil Plotting

Choose any **three** airfoils from the University of Illinois database:

`http://m-selig.ae.illinois.edu/ads/coord_database.html`

Plot the airfoil shapes on top of each other, with the horizontal $\frac{x}{c}$ and vertical $\frac{z}{c}$ directions are nondimensionalized by the airfoil chord length c:

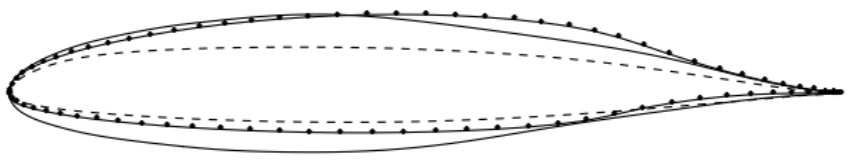

Fig. 4: Three different airfoil cross-sectional geometries

Make **each airfoil unique** with marker and color styles. **Include a legend** that says the name of each airfoil.

NOTE: **Use equal axis scales for all airfoil plots** (i.e. the space between tick marks on the x and y axes is the same). These are geometry plots, so there is no reason to scale the axes differently. You will lose points if you do not do this.

2.2 Cross-Sectional Area via Line Integration

For this problem, we will **calculate the cross-sectional area of a NACA 2412 airfoil** (Located in the file "naca2412_geom.dat"). According to Green's Theorem, the area bounded by a closed contour C can be expressed as line integral around C:

$$Area = \iint_D dA = \frac{1}{2}\oint_C (-ydx + xdy) = \oint_C xdy = \oint_C -ydx$$

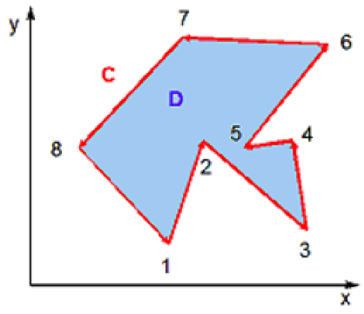

Fig. 5: Integral of Area D may be evaluated equivalently with Green's Theorem about closed contour C (*Source:* `http://www.numericalexpert.com/blog/area_calculation/`)

Since the $\left(\frac{x}{c}, \frac{z}{c}\right)$ coordinates of an airfoil create a continuous, closed contour, we can perform a line integral on the dimensional airfoil coordinates to determine the cross-sectional area:

$$x = \frac{x}{c}\cdot c,\ z = \frac{z}{c}\cdot c$$

$$Area = \oint zdx$$

$$Area = np.trapz(z,x)$$

where chord length $c = 9.5ft$.

3 Airfoil Surface Pressure and Numeric Differentiation

3.1 Airfoil Surface Pressure

When an aerodynamicist is interested in the forces and moments induced on a wing, it can be convenient to know the distribution of pressure over the airfoil cross-section caused by the surrounding flow.

Plot the nondimensional surface pressure coefficient distribution $-C_P$ vs $\frac{x}{c}$ over a NACA 2412 airfoil provided in the file "naca2412_SurfPress_a6.csv". CSV stands for Comma Separated Values, and if you load the file in a text editor, you will see that each data value is separated by a comma. This file can be loaded into Python using Numpy's "loadtxt", specifying the comma delimiter:

$$x, Cpl, Cpu = np.loadtxt(''naca2412\_SurfPress\_a6.csv'', unpack = True, delimiter = ',')$$

where $x = \frac{x}{c}$, and Cpl/Cpu are the lower and upper surface pressure coefficients, respectively, and should be **labeled separately in a legend**. Additionally, you must **reverse the y-axis or plot negative** C_P to provide a more intuitive representation of the pressure's effect (See Fig. 6):

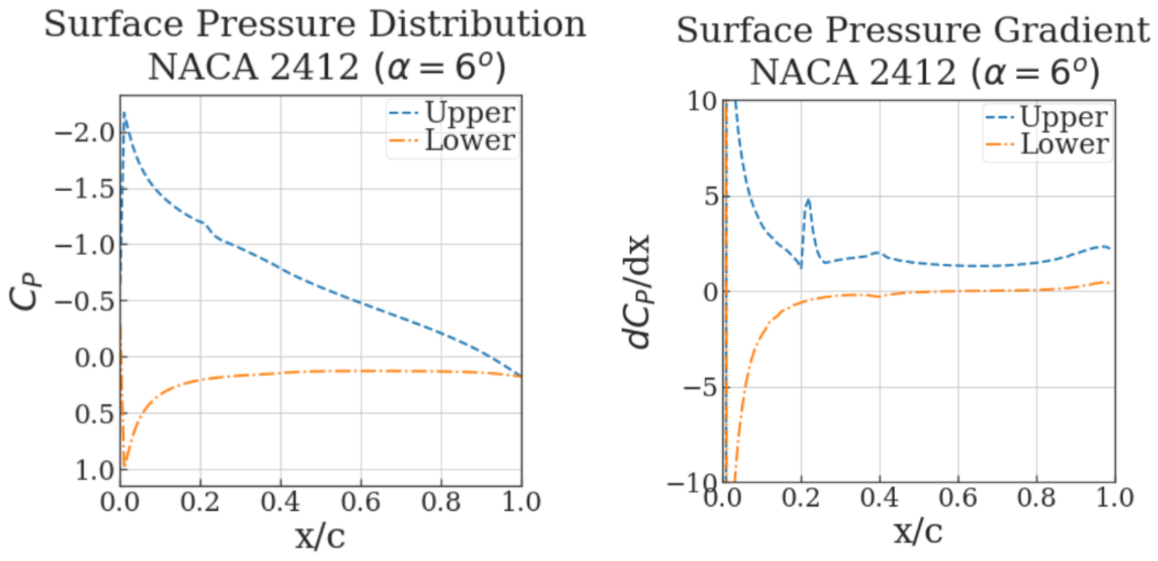

Fig. 6: Example airfoil surface pressure plot (left) and surface pressure gradient plot (right)

3.2 Surface Pressure Gradient and Numeric Differentiation

As we will find out later in the course, the spatial gradient of the surface pressure distribution $\frac{\partial C_P}{\partial x/c}$ is also a very important parameter in determining aerodynamic and stability characteristics of airfoils and wings.

Calculate and plot the surface pressure gradient of the data from the previous problem using first order, forward differentiation:

$$\left(\frac{\partial C_P}{\partial x}\right)_i = \frac{C_{P,i+1} - C_{P,i}}{x_{i+1} - x_i}$$

where x is actually $\frac{x}{c}$ in this case.

4 Lift Curves and Linear Interpolation

4.1 Lift Curve and Excel Files

The plot characterizing airfoil lift performance at various angles of attack is called a lift curve (C_l vs α). **Plot the lift curve** provided in the Excel file “naca2412_LiftCurve.xlsx”. To load the Excel file into Python, you will need to either convert it into a ‘.csv’ file or use Panda’s “read_excel” function (for this to work, you will need to install the “*pandas*” and “*xlrd*” Python modules:

$$
\begin{aligned}
&import\ pandas\ as\ pd\\
&df = pd.read\_excel('Data/naca2412\_LiftCurve.xlsx')\\
&alpha = df['alpha']\\
&Cl = df['Cl']
\end{aligned}
$$

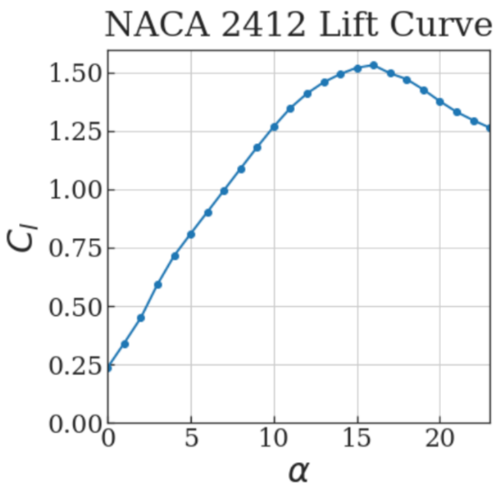

Fig. 7: Example airfoil lift curve

4.2 Linear Interpolation

The data we used for the plot in Fig. 7 is discrete, which means we only have data at certain intervals; in this case, every degree of angle attack. However, when performing detailed aerodynamic analyses, it is necessary to be even more precise than our original data set allows. When our data points are relatively close together, as in this case, we can use simple linear interpolation, where we draw a line between two points and choose a value on this line corresponding to the specific data point we need.

Using Numpy’s “interp” function to **determine the coefficient of lift C_l at an angle of attack** $\alpha = 5.65^o$.

5 Linear Algebra

When solving equations numerically, a domain (e.g. an airfoil or a wing) is decomposed into many discrete sections and the governing equations must be solved at each of these locations. This is accomplished using the principles of linear algebra, so for now, we will practice solving systems of equations in Python.

Solve the following system of equations using Numpy's "linalg.solve" function:

$$\begin{aligned} w+2x+3y+4z &= 12 \\ 3w+2x-2y+3z &= 10 \\ x+y &= -1 \\ 2w+x+y-2z &= -5 \end{aligned}$$

To solve this system for $[w, x, y, z]$, we need to put it into matrix form, as below:

$$\underbrace{\begin{bmatrix} 1 & 2 & 3 & 4 \\ 3 & 2 & -2 & 3 \\ 0 & 1 & 1 & 0 \\ 2 & 1 & 1 & -2 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix}}_{\lambda} = \underbrace{\begin{bmatrix} 12 \\ 10 \\ -1 \\ -5 \end{bmatrix}}_{b}$$

We can then solve the system as such:

$$\begin{aligned} A\lambda &= b \\ \lambda &= A^{-1}b \\ lambda &= np.linalg.solve(A, b) \end{aligned}$$

6 Fluids Review

Start on Fluids Review (self-study Section 2) – material is fair game for the Midterm and Final exams. Complete in time for midterm.