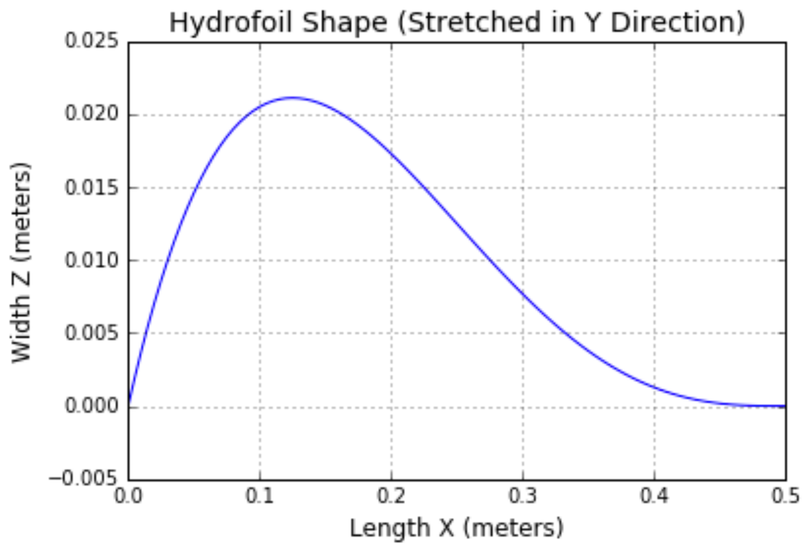
ME560 Assignment 3

Group 17

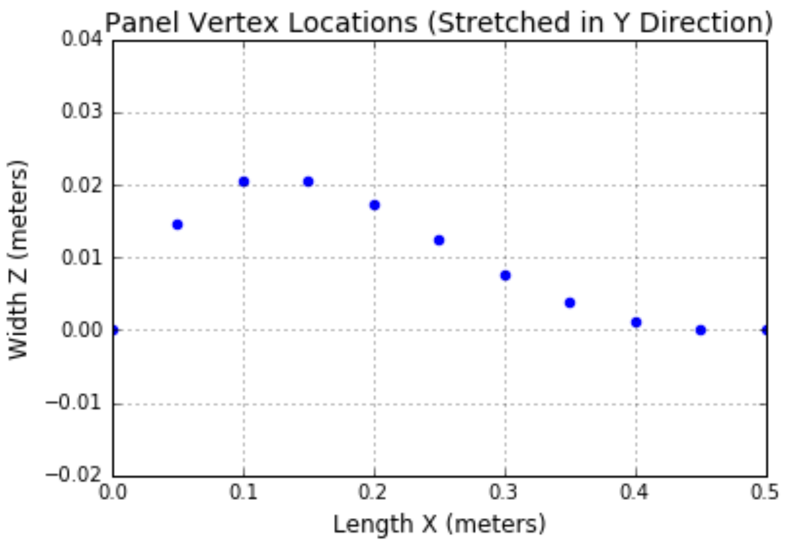
Andrew Alferman
Nathan Schorn

1. Plot the surface profile η vs distance *x*. Describe the shape relative to a typical airfoil you might find.



The above figure was created using the attached Python code and shows the shape of the surface profile, stretched in the *y* direction. The shape is thin and has a long chord length, but is otherwise similar to the top of an airfoil shape, though it is not mirrored across the positive *x* axis.

2. Select flat panels, using at least 10 along the length of the surface. Draw a sketch showing your panel distribution. Create a table listing your (*x, z*) coordinates at the beginning and end of each panel as well as the locations for vortex placement and collocation points along the surface.



Vertex Locations

	x	z
1	0.0000	0.0000
2	0.0500	0.0146
3	0.1000	0.0205
4	0.1500	0.0206
5	0.2000	0.0173
6	0.2500	0.0125
7	0.3000	0.0077
8	0.3500	0.0038
9	0.4000	0.0013
10	0.4500	0.0002
11	0.5000	0.0000

Panel	Collocation Points		Vortex Points	
	x	z	x	z
1	0.0375	0.0109	0.0125	0.0036
2	0.0875	0.0190	0.0625	0.0161
3	0.1375	0.0206	0.1125	0.0205
4	0.1875	0.0181	0.1625	0.0198
5	0.2375	0.0137	0.2125	0.0161
6	0.2875	0.0089	0.2625	0.0113
7	0.3375	0.0048	0.3125	0.0067
8	0.3875	0.0019	0.3625	0.0032
9	0.4375	0.0005	0.4125	0.0010
10	0.4875	0.0000	0.4625	0.0001

The points were selected using the attached Python code by evenly spacing 11 points from x = 0 to x = 0.5, inclusive of 0 and 0.5, to create 10 panels. The z value was found using the given formula. The vortex locations were placed at L = ¼ for each panel, and the collocation points were placed at L = ¾ for each panel.

3. Evaluate the matrix a_{ij} (as defined in class which is the matrix for the condition of vortex circulation values = 1.0) and put the results in the form of a table for each angle of attack.

The a_{ij} matrix was evaluated in the attached Python code by first finding the u and w values of flow velocity at each point. The formula were derived in class and can be found in the online handout (Additional Notes on the Panel Method, available on Canvas) as follows:

$$u = \frac{\Gamma}{2\pi}\,\frac{(z-z_0)}{(x-x_0)^2+(z-z_0)^2}$$

$$w = \frac{-\Gamma}{2\pi}\,\frac{(x-x_0)}{(x-x_0)^2+(z-z_0)^2}$$

These formula were evaluated at each collocation point using the distance between each collocation point and each vortex point. Gamma was equal to one in each instance for the purpose of creating the a_{ij} matrix. Next, the u and w values were placed in a vector q, and the dot product of that vector and a vector n denoting the direction normal to each panel was obtained. Because this process assumes that the circulation at each panel is equal to one, the uniform flow velocity U was not taken into account, and thus the angle of attack was not considered. Therefore, there is only a single a_{ij} matrix for any angle of attack. The 10 x 10 matrix, with i values increasing horizontally and j values increasing downwards is as follows:

a_{ij}:	1	2	3	4	5	6	7	8	9	10
1	-6.11	6.22	2.08	1.24	0.88	0.68	0.55	0.47	0.40	0.36
2	-2.07	-6.32	6.34	2.11	1.26	0.90	0.70	0.57	0.48	0.42
3	-1.25	-2.11	-6.37	6.36	2.11	1.27	0.90	0.70	0.58	0.49
4	-0.90	-1.27	-2.12	-6.35	6.34	2.11	1.27	0.91	0.71	0.58
5	-0.70	-0.91	-1.27	-2.11	-6.34	6.34	2.11	1.27	0.91	0.71
6	-0.57	-0.71	-0.91	-1.27	-2.11	-6.34	6.34	2.12	1.27	0.91
7	-0.49	-0.58	-0.71	-0.91	-1.27	-2.11	-6.35	6.35	2.12	1.27
8	-0.42	-0.49	-0.58	-0.70	-0.91	-1.27	-2.12	-6.36	6.36	2.12
9	-0.37	-0.42	-0.49	-0.58	-0.70	-0.91	-1.27	-2.12	-6.36	6.37
10	-0.34	-0.37	-0.42	-0.49	-0.58	-0.71	-0.91	-1.27	-2.12	-6.37

Note: Indices of i and j are included in the first row and column of the above table, respectively.

4. Obtain the distribution of Γ, that is determine Γ for each panel, and plot this versus x for each α. Show all work.

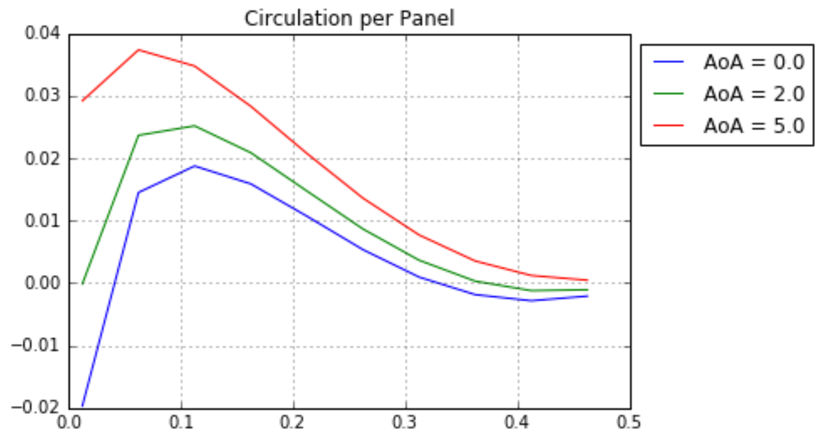
The value of Γ for each panel was found using the attached Python code by first obtaining the dot product of x and z components of U for each angle of attack with the x and z components of a unit vector normal to the surface of each panel (named ni in the attached code):

```
for i in range(len(ni)):
        umatrix.append(-1 * np.dot([u * np.cos(np.radians(aoa)),
                                    u * np.sin(np.radians(aoa))],
                                   [np.cos(ni[i]), np.sin(ni[i])]))
```

Next, the dot product between the inverse of the a_{ij} matrix and the previously obtained vector to obtain a vector of Γ values for each panel:

```
    aiji = np.linalg.inv(aij)
    gammas = np.dot(aiji, umatrix)
    return gammas
```

The value of Γ on each panel can be found in the figure and table below for each angle of attack α. The plot shows that the circulation decreases across the length of the hydrofoil after the first few panels.



AoA	Panel 1	Panel 2	Panel 3	Panel 4	Panel 5	Panel 6	Panel 7	Panel 8	Panel 9	Panel 10
0	-0.0197	0.0145	0.0187	0.0159	0.0107	0.0053	0.0009	-0.0019	-0.0028	-0.0021
2	-0.0001	0.0237	0.0252	0.0209	0.0147	0.0087	0.0037	0.0003	-0.0012	-0.0011
5	0.0292	0.0373	0.0347	0.0283	0.0208	0.0136	0.0077	0.0035	0.0012	0.0005

5. Find the local pressure difference across the airfoil, Δp_k, for each element, k, based on the local lift force, L_k, and plot this versus *x* for each α in terms of the local pressure coefficient, ΔC_p. Show all work.

The pressure difference across each panel Δp_k was found by dividing the local lift of each panel per 1m of span length by the width of each panel. The local lift force was found using the formula $L_k' = \rho U_\infty \Gamma_k$ where L_k' is the lift per 1m span length for each panel, U_∞ is the uniform flow velocity, ρ is the density of the fluid, and Γ_k is the circulation for each panel which was obtained in step 4. The local pressure coefficient was then found using the formula:

$$c_p = \frac{P - P_0}{\frac{1}{2}\rho U_\infty^2}$$

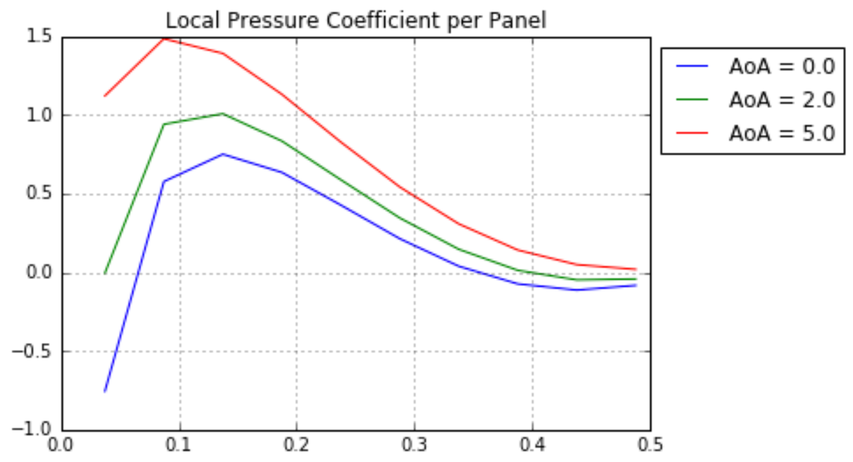
Where P_0 is the pressure far away from the hydrofoil and P is the pressure at the point to be evaluated, thus $P - P_0$ is equal to Δp_k. Combining this formula with $L_k' = \rho U_\infty \Gamma_k$ and $\Delta p_k = L_k'/w_k$ where w_k is the width of each panel, we get:

$$c_p = \frac{2\Gamma_k}{U_\infty w_k}$$

This c_p value was calculated at each panel:

```
for k in range(len(gammas)):
    cp.append(2 * gammas[k] / (u * plm[k]))
```

Which produces the following figure:



6. Determine the total lift force on the surface for each α. Compare this to the expected lift coefficient for a two dimensional flat plate at the same angle of attack.

The total lift coefficients were computed in the attached Python code by adding together the circulation from each panel, then multiplying by the uniform flow velocity U, and the density ρ (1000 kg/m^3) using the equation $L' = \rho U \Gamma$ as follows:

```
    for j in range(len(gammas)):
        gammat += gammas[j]
    tliftm.append(gammat * u * 1000)
```

The angle of attack was taken into account in each case when the Γ value of each plate was computed, as discussed in step 4 above. Using this formula, the attached Python code produced the following output:

```
For 1m Span Length:
Lift with AoA = 0.0: 39.64
Flat Plate Lift with AoA = 0.0: 0.00
Lift with AoA = 2.0: 94.60
Flat Plate Lift with AoA = 2.0: 54.82
Lift with AoA = 5.0: 176.80
Flat Plate Lift with AoA = 5.0: 136.90
```

The output demonstrates that the hydrofoil produces approximately 40 N of lift force per 1m span length greater than a two dimensional flat plate for the same angle of attack, thus the hydrofoil shape does have some effect on lift.

7. Now to get a feel for what the influence of viscous forces may be compare these results to flow over a flat plate of length *c*, for zero angle of attack. Look up the solution for the viscous force on the surface (assume boundary layer flow along the length c) and compare this in magnitude to the lift force at zero angle of attack. Indicate the conditions and equations that you use.

This problem was clarified as follows:
You are to compare the lift force at zero angle of attack of the surface described by "eta(x)" to the viscous force predicted for flow over a flat plate which is also at zero angle of attack. Discuss the relative magnitudes.

The viscous force discussed is the drag force. The drag force can be found using an equation on page 221 of Nuun:

$$C_f = \frac{D_f}{\frac{1}{2}\rho U_\infty^2 bL}$$

Where D_f is the drag force, b is the plate width (span length), L is the plate length (chord length), and C_f is as follows:

$$C_f = \frac{1.328}{Re_L^{1/2}}$$

Where the Reynolds number Re_L is defined as $Re = U_\infty L/\upsilon$, with υ equal to the kinematic viscosity, equal to 1.003×10^{-6} m^2/sec for water at 20 degrees C. Combining these two formulas and substituting in values, we find that the drag force is 0.47 N per 1m of span length. This is compared to a lift force of approximately 40 N per 1m of span length for the hydrofoil evaluated in this assignment. The hydrofoil, being a relatively flat, very thin shape, will have a drag force roughly similar to the drag force of the flat plate. The lift will be roughly 2 orders of magnitude greater than the drag force in this case, which is an excellent lift/drag ratio.

Below is the code used to obtain the solutions to all of the above problems:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Oct 31 11:47:02 2016

@author: Andrew Alferman and Nathan Schorn
"""

import numpy as np
import matplotlib.pyplot as plt

# Global variables
c = 0.5
meshpoints = 10
aoam = [0.0, 2.0, 5.0]
u = 1.0
p_inf = 101000
method = 1


# Create a function that uses the given equation to make vertexes for the
# panels. The function also finds the vortex points, colocation points, and
# angle outward (normal) from the panel.
def createaij(meshpoints, c):
    # Setting up lists for X coordinate (xc), eta coordinate (etac), vortex
    # coordinates (vcx/vcz), colocation point coordinates (clcx/clcz),
    # and angle normal to panel (ni)
    xc, etac, vcx, vcz, clcx, clcz, ni, plm = [], [], [], [], [], [], [], []
    # Initialize a few variables to create the points.
    x = 0
    dx = c / meshpoints
    c_i = 1 / c
    # Create all of the vertex points.
    for i in range(meshpoints + 1):
        xc.append(x)
        etac.append(0.2 * x * c_i * (1 - x*c_i)**3)
        x += dx
    # Create the vortex points, colocation points, normal angles, and lengths
    for i in range(1, len(xc)):
        vcx.append(xc[i] - 0.75*(xc[i]-xc[i-1]))
        vcz.append(etac[i] - 0.75*(etac[i]-etac[i-1]))
        clcx.append(xc[i] - 0.25*(xc[i]-xc[i-1]))
        clcz.append(etac[i] - 0.25*(etac[i]-etac[i-1]))
        ni.append(np.pi/2 + np.arctan((etac[i]-etac[i-1])/(xc[i]-xc[i-1])))
        plm.append(np.sqrt((xc[i]-xc[i-1])**2 + (etac[i]-etac[i-1])**2))
    # Create the aij matrix
    aij = []
    for i in range(len(vcx)):
        aijr = []
        for j in range(len(clcx)):
            lr = (clcx[i] - vcx[j])**2 + (clcz[i] - vcz[j])**2
            u = (1 / (2 * np.pi)) * (clcz[i] - vcz[j]) / lr
            w = (-1 / (2 * np.pi)) * (clcx[i] - vcx[j]) / lr
            n = [np.cos(ni[i]), np.sin(ni[i])]
            q = [u, w]
            aijr.append(np.dot(q, n))
        aij.append(aijr)
    aij = np.array(aij)
    return xc, etac, vcx, vcz, clcx, clcz, ni, aij, plm


# Create a function that finds the circulation across all the panels
```

```
def findgammas(ni, aij, aoa, u):
    umatrix = []
    for i in range(len(ni)):
        umatrix.append(-1 * np.dot([u * np.cos(np.radians(aoa)),
                                    u * np.sin(np.radians(aoa))],
                                   [np.cos(ni[i]), np.sin(ni[i])]))
    aiji = np.linalg.inv(aij)
    gammas = np.dot(aiji, umatrix)
    return gammas


def flatplate(meshpoints, c):
    # Setting up lists for X coordinate (xc), eta coordinate (etac), vortex
    # coordinates (vcx/vcz), colocation point coordinates (clcx/clcz),
    # and angle normal to panel (ni)
    xc, etac, vcx, vcz, clcx, clcz, ni, plm = [], [], [], [], [], [], [], []
    # Initialize a few variables to create the points.
    x = 0
    dx = c / meshpoints
    # Create all of the vertex points.
    for i in range(meshpoints + 1):
        xc.append(x)
        etac.append(0)
        x += dx
    # Create the vortex points, colocation points, normal angles, and lengths
    for i in range(1, len(xc)):
        vcx.append(xc[i] - 0.75*(xc[i]-xc[i-1]))
        vcz.append(0)
        clcx.append(xc[i] - 0.25*(xc[i]-xc[i-1]))
        clcz.append(0)
        ni.append(np.pi/2)
        plm.append(0.05)
    # Create the aij matrix
    aij = []
    for i in range(len(vcx)):
        aijr = []
        for j in range(len(clcx)):
            lr = (clcx[i] - vcx[j])**2
            u = 0
            w = (-1 / (2 * np.pi)) * (clcx[i] - vcx[j]) / lr
            n = [0, 1]
            q = [u, w]
            aijr.append(np.dot(q, n))
        aij.append(aijr)
    aij = np.array(aij)
    return xc, etac, vcx, vcz, clcx, clcz, ni, aij, plm


xc, etac, vcx, vcz, clcx, clcz, ni, aij, plm = createaij(meshpoints, c)
fxc, fetac, fvcx, fvcz, fclcx, fclcz, fni, faij, fplm =\
    flatplate(meshpoints, c)
tliftm, gammam, cpm = [], [], []
ftliftm, fgammam, fcpm = [], [], []
for i in aoam:
    cp = []
    fcp = []
    gammas = findgammas(ni, aij, i, u)
    gammam.append(np.ndarray.tolist(gammas))
    gammat = 0
    fgammas = findgammas(fni, faij, i, u)
    fgammam.append(np.ndarray.tolist(fgammas))
    fgammat = 0
    for j in range(len(gammas)):
        gammat += gammas[j]
        fgammat += fgammas[j]
    tliftm.append(gammat * u * 1000)
    ftliftm.append(fgammat * u * 1000)
    for k in range(len(gammas)):
        cp.append(2 * gammas[k] / (u * plm[k]))
        fcp.append(2 * fgammas[k] / (u * fplm[k]))
    cpm.append(cp)
    fcpm.append(fcp)
    plt.figure(1)
    plt.title('Circulation per Panel')
    plt.plot(vcx, gammas, label='AoA = {}'.format(i))
    plt.legend(bbox_to_anchor=(1, 1), loc=2)

    plt.figure(2)
    plt.title('Local Pressure Coefficient per Panel')
    plt.plot(clcx, cp, label='AoA = {}'.format(i))
    plt.legend(bbox_to_anchor=(1, 1), loc=2)
gammam = np.array(gammam)
fgammam = np.array(fgammam)

plt.figure(3)
plt.scatter(xc, etac, color='blue')
plt.xlabel('Length X (meters)', fontsize=12)
plt.ylabel('Width Z (meters)', fontsize=12)
plt.title('Panel Vertex Locations (Stretched in Y Direction)', fontsize=14)
plt.xlim(0, 0.5)

plt.figure(4)
plt.plot(xc, etac, color='blue')
plt.xlabel('Length X (meters)', fontsize=12)
plt.ylabel('Width Z (meters)', fontsize=12)
plt.title('Hydrofoil Shape (Stretched in Y Direction)', fontsize=14)
plt.xlim(0, 0.5)

for i in range(1, 5):
    plt.figure(i)
    plt.grid(b=True, which='both')
plt.show()

print('For 1m Span Length:')
for i in range(len(aoam)):
    print('Lift with AoA = {}: {:.2f}'.format(aoam[i], tliftm[i]))
    print('Flat Plate Lift with AoA = {}: {:.2f}'.format(aoam[i], ftliftm[i]))

# The code below just outputs the location of points in an easy to read format
vertexes, clc, vc = [], [], []
for i in range(len(clcx)):
    clc.append([clcx[i], clcz[i]])
    vc.append([vcx[i], vcz[i]])
for i in range(len(xc)):
    vertexes.append([xc[i], etac[i]])
vertexes = np.array(vertexes)
clc = np.array(clc)
vc = np.array(vc)
```