HW #3 (Source Panel Method)

David Tran (ID: 205-492-874)

April 19, 2021

Results and Discussion

As reference, the following equations were used in the MATLAB script:

$$\begin{aligned} I_{ij} &= \int_j \frac{\partial}{\partial n_i} \left(\ln r_{ij}\right) ds_j \\ &= \int_0^{S_j} \frac{Cs_j + D}{s_j^2 + 2As_j + B} ds_j \\ &= \frac{C}{2} \ln\left(\frac{S_j^2 + 2AS_j + B}{B}\right) + \frac{D - AC}{E}\left(\tan^{-1}\frac{S_j + A}{E} - \tan^{-1}\frac{A}{E}\right) \end{aligned} \tag{1}$$

where the constants A, B, C, D, E are

$$\begin{aligned} A &= -\left(x_i - X_j\right)\cos\Phi_j - \left(y_i - Y_j\right)\sin\Phi_j \\ B &= \left(x_i - X_j\right)^2 + \left(y_i - Y_j\right)^2 \\ C &= \sin\left(\Phi_i - \Phi_j\right) \\ D &= \left(y_i - Y_j\right)\cos\Phi_i - \left(x_i - X_j\right)\sin\Phi_i \\ E &= \sqrt{B - A^2} \\ S_j &= \sqrt{\left(X_{j+1} - X_j\right)^2 + \left(Y_{j+1} - Y_j\right)^2} \end{aligned}$$

For velocity calculations, the quantity J_{ij} is desired,

$$\begin{aligned} J_{ij} &= \int_j \frac{\partial}{\partial s} \left(r_{ij}\right) ds_j \\ &= \frac{D - AC}{2E} \ln\frac{S_j^2 + 2AS_j + B}{B} - C\left(\tan^{-1}\frac{S_j + A}{E} - \tan^{-1}\frac{A}{E}\right) \end{aligned} \tag{2}$$

To verify that the generated results are correct, we must verify that the sum of all sources/-sources must obey the following relation:

$$\sum_{j=1}^{N} \lambda_j S_j = 0 \tag{3}$$

The pressure coefficient $C_{p,i}$ is

$$C_{p,i} = 1 - \left(\frac{V_i}{V_\infty}\right)^2 \tag{4}$$

Also, the equation for the points along the NACA 0012 airfoil is

$$y = \frac{t}{0.2}c\left[0.2969\sqrt{\frac{x}{c}} - 0.1260\left(\frac{x}{c}\right) - 0.3516\left(\frac{x}{c}\right)^2 + 0.2843\left(\frac{x}{c}\right)^3 - 0.1015\left(\frac{x}{c}\right)^4\right] \quad (5)$$

For $n = 50$ panels, we find that the sum of all sources/sinks multiplied with the length of its respective panel, which is defined by equation (3), is

```
Sum of all sources/sinks for 50 panels:  0.005661
```

The airfoil geometry with $n = 50$ panels superimposed above the airfoil plot which contains 1000 points whose points are found by equation (5).

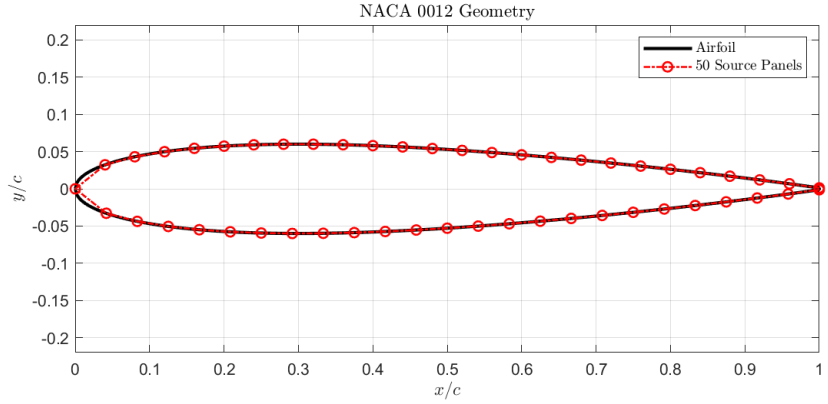

Figure 1: NACA 0012 airfoil with $n = 50$ panels superimposed on it

After performing the calculations, the following C_p vs. x/c plot was obtained:

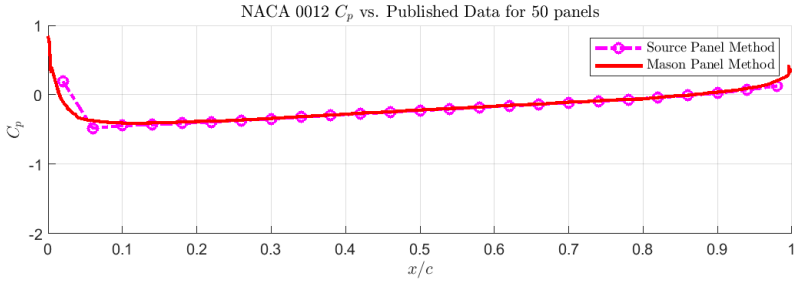

Figure 2: C_p vs. x/c for $n = 50$ panels compared to the Mason panel method

Results for C_p vs. x/c compared to the Gregory and O'Reilly 1970 experiments are shown below in figure (3).

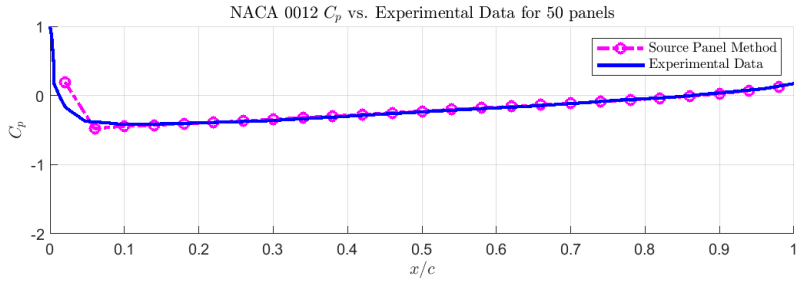

Figure 3: Generated results versus experimental data from the 1970 Gregory and O'Reilly experiments

We can see that there is a slight disparity near the trailing and leading edges ($x/c < 0.1$ and $x/c > 0.9$) between the results I generated and the Mason panel method and the experimental data. However, the results are very closely aligned everywhere in between the aforementioned region despite a relatively small amount of panels. Increasing n to $n = 100$ panels, the sum of all sources/sinks is approximately half that of the value found for 50 panels:

```
Sum of all sources/sinks for 100 panels:  0.003963
```

This is an unsurprising result, as we expect this value to get smaller and smaller as the number of panels increases. Plotting the airfoil using equation (5) with $n = 100$ panels, we find a better fit with the actual airfoil's shape:

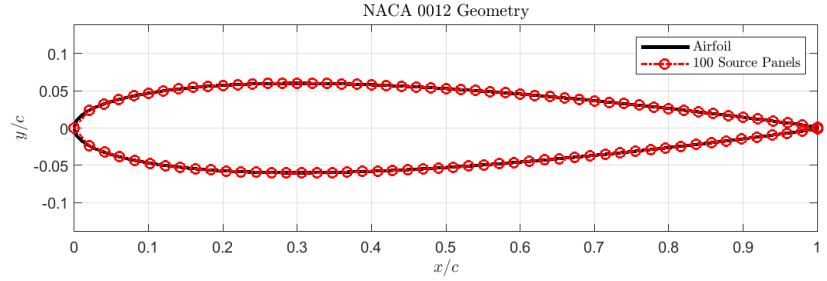

Figure 4: NACA 0012 airfoil with $n = 100$ panels, showcasing a better approximation

The calculated results compared to the Mason source panel method for 100 panels is shown in figure (5).

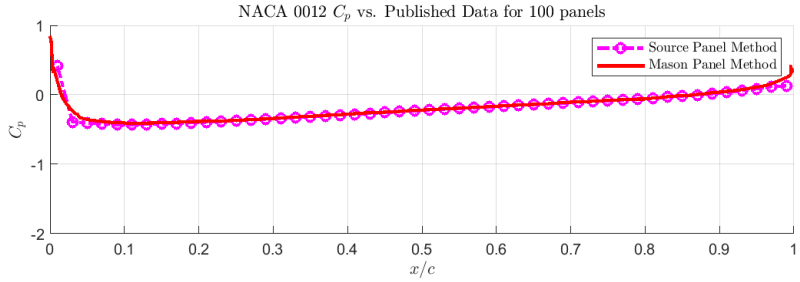

Figure 5: C_p vs. x/c for the conventional source panel versus Mason's method

We can see a better fit with Mason's panel method, especially in the leading edge area. However, the results are slightly off, but Mason's method utilizes approximately two times the amount of points, so this is an expected disparity. Another interesting observation is that there seems to be a greater gap at the last point in comparison to a smaller number of panels. Compared to the O'Reilly experiments, the data is also very closely aligned:

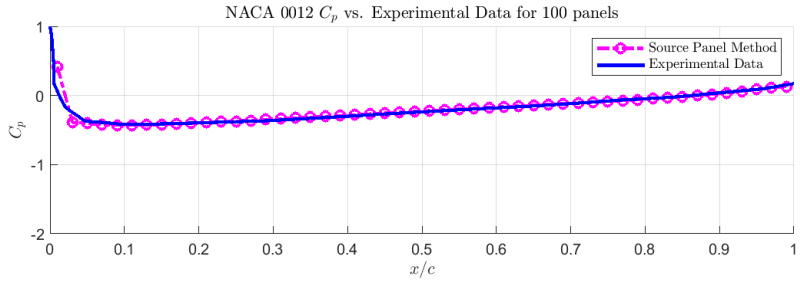

Figure 6: Comparison to the O'Reilly experiments for $n = 100$ panels

For $n = 200$ panels, the sum is slightly more than half that of the value found for 100 panels:

```
Sum of all sources/sinks for 200 panels:  0.002456
```

Once again, this sum decreases, indicating that the source strengths of each panel $(\lambda_1, \ldots, \lambda_N)$ are correct for that given number of panels. The geometry of $n = 200$ panels is obviously more closely aligned to the true airfoil shape.

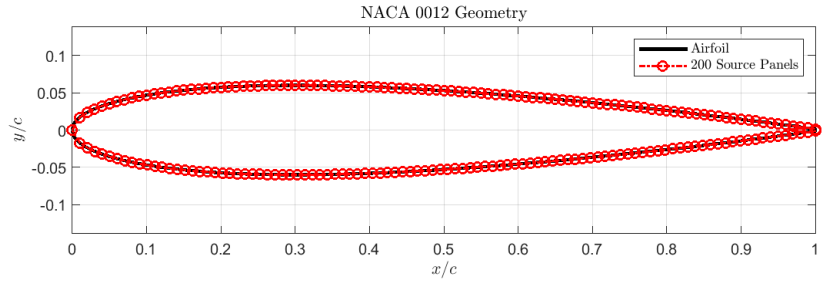

Figure 7: NACA 0012 airfoil with $n = 200$ panels

Plotting the source panel data versus Mason's data, we find an ever closer approximation to their method.

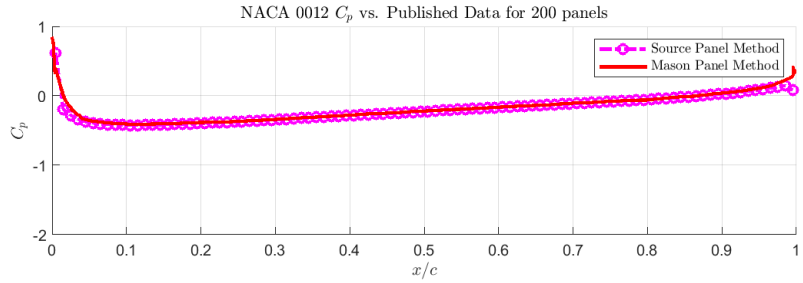

Figure 8: C_p vs. x/c for $n = 200$ panels compared to the Mason panel method

One surprising result is that this final endpoint essentially diverges as one increases n; indeed, comparing these results to figures (2) and (5), a divergence becomes readily apparent and must be resolved in future renditions. This disparity is also noticeable when plotted with the experimental data, but this disagreement was not as noticeable in figures (3) and (6).

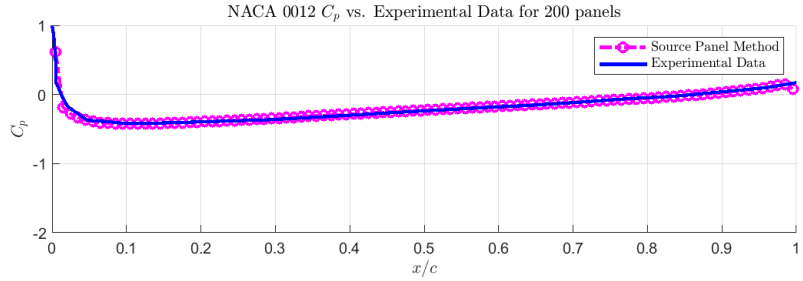

Figure 9: Comparison to experimental data for $n = 200$ panels

Using equation (3), we find that the sum is halved for $n = 400$ panels.

```
Sum of all sources/sinks for 400 panels:  0.001511
```

Doubling the panels to $n = 400$ panels, the geometry of the panels is shown in figure (10).

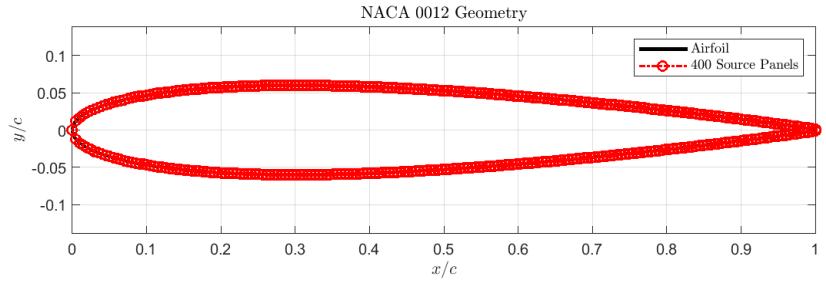

Figure 10: Geometry of $n = 400$ panels for a NACA 0012 airfoil

The Mason panel method comparison results once again demonstrate this final point disparity, but even more prominently so in comparison to figures (2), (5), and (8). However, the rest of the results are essentially indistinguishable from one another when compared especially to figure (2).

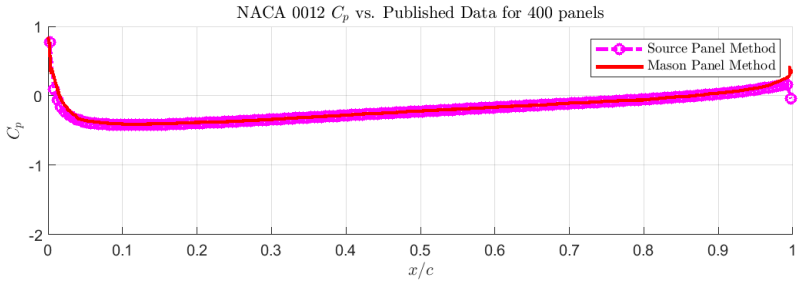

Figure 11: C_p vs. x/c for $n = 400$ panels to the Mason panel method

As aforementioned, the last control point is a point of contention, growing even more in this final set of points. Finally, for the experimental data, we find a similar observation:

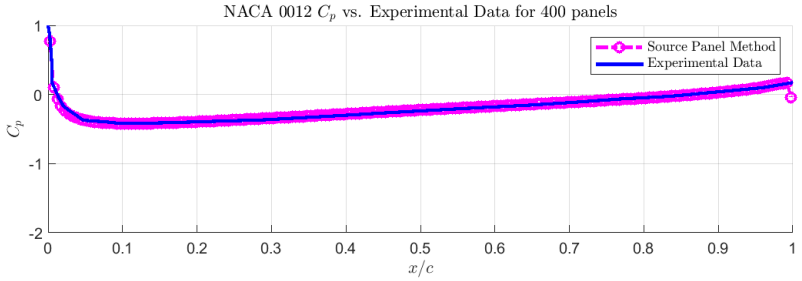

Figure 12: Experimental data compared to the source panel method for $n = 400$

We find that the effect of increasing the number of panels enhances the accuracy of the data, as evident by the decreasing sum of source/sink strengths and the closer adherence to experimental and more robust programs. As stated, remedying this last control point disparity is a strong contender for future improvements. Indeed, it almost appears discontinuous and out-of-place with the rest of the calculated C_p points. For further inspection, the MATLAB code is shown below in Appendix A. Please note that the actual .m file does not contain separate .m files for the functions, as I found that it was too cumbersome, especially for the graders to have to download all of the files along with increasing the chances of errors running the script if a file is not submitted by accident.

A MATLAB Source Code

```
%
    ================================================================================

%Source Panel Method
%David Tran
%UCLA ID: 205-492-874
%{
Description:
This script seeks to implement the source panel method to generate a
    plot
```

```
of the pressure coefficient Cp as a function of x/c, where c is the
    chord
length. Results from this method will be compared to provided
    experimental
and computational results to assess its accuracy. Further, data will be
created for panel counts of 50, 100, 200, and 400 to demonstrate the
    effect
of increasing the number of panels.
%}

%==============================Clear Cache
    ==================================
clc; close all; clear all;

%==============================Constants
    ====================================
alpha = 0;                                  %Angle of attack in degrees
N_1 = 50;                                   %Number of panels for set 1
N_2 = 100;                                  %Number of panels for set 2
N_3 = 200;                                  %Number of panels for set 3
N_4 = 400;                                  %Number of panels for set 4
V_infty = 1;                                %speed of freestream velocity
    in [m/s]
c = 1;                                      %chord length in [m]

%Plot the airfoil
[X, Y] = plotAirfoil();

%%First set of data points for N = 50 panels
A1 = zeros(N_1, N_1);
A1_t = zeros(N_1, N_1);
B1 = zeros(N_1, 1);
B1_t = zeros(N_1, 1);
beta_i1 = zeros(1, N_1);
phi_i1 = zeros(1, N_1);
Sj_1 = zeros(1, N_1);
[X1, Y1] = panels(N_1);                      %Obtain panel endpoints
[xc, yc] = controlPoint(N_1, X1, Y1);        %Obtain panel control points

%Populate the A and B matrices
for i=1:N_1
    phi_i1(i) = phi_calc(X1(i), Y1(i), X1(i+1), Y1(i+1));
    beta_i1(i) = phi_i1(i) + pi/2;
    B1(i) = -V_infty * cos(beta_i1(i));
    Sj_1(i) = sqrt((X1(i+1) - X1(i))^2 + (Y1(i+1) - Y1(i))^2);
    B1_t(i) = V_infty * sin(beta_i1(i));
    for j=1:N_1
        if i ~= j
            phi_i1(j) = phi_calc(X1(j), Y1(j), X1(j+1), Y1(j+1));
            A1(i,j) = 1 / (2*pi) * integral(xc(i), yc(i), X1(j), Y1(j),
                X1(j+1), Y1(j+1), phi_i1(i), phi_i1(j));
            A1_t(i, j) = 1 / (2*pi) * integral_j(xc(i), yc(i), X1(j),
               Y1(j), X1(j+1), Y1(j+1), phi_i1(i), phi_i1(j));
        elseif i == j
```

```
            A1(i, j) = 0.5;
            A1_t(i, j) = 0;
        end
    end
end

%Solve the Ax = B system and verify that Eq. 3.157 holds (sum of i-th
%source/sink multiplied by i-th panel length for all panels is 0)
lambda_1 = A1\B1;
sourceSum(lambda_1, Sj_1, N_1);

%Obtain the tangential velocity
V_i1 = tangentialVelocity(A1_t, lambda_1, B1_t, N_1);

%Calcuclate the pressure coefficient
Cp_1 = Cp(V_i1, V_infty, N_1);

%%Second set of data points for N = 100 panels
A2 = zeros(N_2, N_2);
A2_t = zeros(N_2, N_2);
B2 = zeros(N_2, 1);
B2_t = zeros(N_2, 1);
beta_i2 = zeros(1, N_2);
phi_i2 = zeros(1, N_2);
Sj_2 = zeros(1, N_2);
[X2, Y2] = panels(N_2);                    %Obtain panel endpoints
[xc2, yc2] = controlPoint(N_2, X2, Y2);    %Obtain panel control points

%Populate the A and B matrices
for i=1:N_2
    phi_i2(i) = phi_calc(X2(i), Y2(i), X2(i+1), Y2(i+1));
    beta_i2(i) = phi_i2(i) + pi/2;
    B2(i) = -V_infty * cos(beta_i2(i));
    Sj_2(i) = sqrt((X2(i+1) - X2(i))^2 + (Y2(i+1) - Y2(i))^2);
    B2_t(i) = V_infty * sin(beta_i2(i));
    for j=1:N_2
        if i ~= j
            phi_i2(j) = phi_calc(X2(j), Y2(j), X2(j+1), Y2(j+1));
            A2(i,j) = 1 / (2*pi) * integral(xc2(i), yc2(i), X2(j), Y2(j
                ), X2(j+1), Y2(j+1), phi_i2(i), phi_i2(j));
            A2_t(i, j) = 1 / (2*pi) * integral_j(xc2(i), yc2(i), X2(j),
                 Y2(j), X2(j+1), Y2(j+1), phi_i2(i), phi_i2(j));
        elseif i == j
            A2(i, j) = 0.5;
            A2_t(i, j) = 0;
        end
    end
end

%Solve the Ax = B system and verify that Eq. 3.157 holds (sum of i-th
%source/sink multiplied by i-th panel length for all panels is 0)
lambda_2 = A2\B2;
sourceSum(lambda_2, Sj_2, N_2);
```

```
%Obtain the tangential velocity
V_i2 = tangentialVelocity(A2_t, lambda_2, B2_t, N_2);

%Calcuclate the pressure coefficient
Cp_2 = Cp(V_i2, V_infty, N_2);

%%Third set of data points for N = 200
A3 = zeros(N_3, N_3);
A3_t = zeros(N_3, N_3);
B3 = zeros(N_3, 1);
B3_t = zeros(N_3, 1);
beta_i3 = zeros(1, N_3);
phi_i3 = zeros(1, N_3);
Sj_3 = zeros(1, N_3);
[X3, Y3] = panels(N_3);                    %Obtain panel endpoints
[xc3, yc3] = controlPoint(N_3, X3, Y3);    %Obtain panel control points

%Populate the A and B matrices
for i=1:N_3
    phi_i3(i) = phi_calc(X3(i), Y3(i), X3(i+1), Y3(i+1));
    beta_i3(i) = phi_i3(i) + pi/2;
    B3(i) = -V_infty * cos(beta_i3(i));
    Sj_3(i) = sqrt((X3(i+1) - X3(i))^2 + (Y3(i+1) - Y3(i))^2);
    B3_t(i) = V_infty * sin(beta_i3(i));
    for j=1:N_3
        if i ~= j
            phi_i3(j) = phi_calc(X3(j), Y3(j), X3(j+1), Y3(j+1));
            A3(i,j) = 1 / (2*pi) * integral(xc3(i), yc3(i), X3(j), Y3(j
                ), X3(j+1), Y3(j+1), phi_i3(i), phi_i3(j));
            A3_t(i, j) = 1 / (2*pi) * integral_j(xc3(i), yc3(i), X3(j),
                 Y3(j), X3(j+1), Y3(j+1), phi_i3(i), phi_i3(j));
        elseif i == j
            A3(i, j) = 0.5;
            A3_t(i, j) = 0;
        end
    end
end

%Solve the Ax = B system and verify that Eq. 3.157 holds (sum of i-th
%source/sink multiplied by i-th panel length for all panels is 0)
lambda_3 = A3\B3;
sourceSum(lambda_3, Sj_3, N_3);

%Obtain the tangential velocity
V_i3 = tangentialVelocity(A3_t, lambda_3, B3_t, N_3);

%Calcuclate the pressure coefficient
Cp_3 = Cp(V_i3, V_infty, N_3);

%%Fourth set of data points for N = 400 panels
A4 = zeros(N_4, N_4);
A4_t = zeros(N_4, N_4);
B4 = zeros(N_4, 1);
B4_t = zeros(N_4, 1);
```

```
beta_i4 = zeros(1, N_4);
phi_i4 = zeros(1, N_4);
Sj_4 = zeros(1, N_4);
[X4, Y4] = panels(N_4);                    %Obtain panel endpoints
[xc4, yc4] = controlPoint(N_4, X4, Y4);    %Obtain panel control points

%Populate the A and B matrices
for i=1:N_4
    phi_i4(i) = phi_calc(X4(i), Y4(i), X4(i+1), Y4(i+1));
    beta_i4(i) = phi_i4(i) + pi/2;
    B4(i) = -V_infty * cos(beta_i4(i));
    Sj_4(i) = sqrt((X4(i+1) - X4(i))^2 + (Y4(i+1) - Y4(i))^2);
    B4_t(i) = V_infty * sin(beta_i4(i));
    for j=1:N_4
        if i ~= j
            phi_i4(j) = phi_calc(X4(j), Y4(j), X4(j+1), Y4(j+1));
            A4(i,j) = 1 / (2*pi) * integral(xc4(i), yc4(i), X4(j), Y4(j
                ), X4(j+1), Y4(j+1), phi_i4(i), phi_i4(j));
            A4_t(i, j) = 1 / (2*pi) * integral_j(xc4(i), yc4(i), X4(j),
                Y4(j), X4(j+1), Y4(j+1), phi_i4(i), phi_i4(j));
        elseif i == j
            A4(i, j) = 0.5;
            A4_t(i, j) = 0;
        end
    end
end

%Solve the Ax = B system and verify that Eq. 3.157 holds (sum of i-th
%source/sink multiplied by i-th panel length for all panels is 0)
lambda_4 = A4\B4;
sourceSum(lambda_4, Sj_4, N_4);

%Obtain the tangential velocity
V_i4 = tangentialVelocity(A4_t, lambda_4, B4_t, N_4);

%Calcuclate the pressure coefficient
Cp_4 = Cp(V_i4, V_infty, N_4);

%================================Plots
    ====================================
%Experimental and Computational Data Files
Cp_comp = importdata("Cpsurf_Mason.dat");
xread = Cp_comp(:,1);
Cpread = Cp_comp(:,2);

Cp_comp2 = importdata("Cpsurf_GregoryOReilly.dat");
xread2 = Cp_comp2(:,1);
Cpread2 = Cp_comp2(:,2);

%Plots for N = 50
set(gcf, 'Position',  [400, 100, 800, 800]);
figure(1);
subplot(3,1,1);
plot(X, Y, 'k', 'LineWidth', 2);
```

```
hold on;
plot(X1, Y1, '-.or', 'LineWidth', 1.25);
title('NACA 0012 Geometry', 'Interpreter', 'latex');
xlabel('$x/c$', 'Interpreter', 'latex');
ylabel('$y/c$', 'Interpreter', 'latex');
N1_legend = (sprintf('%d Source Panels', N_1));
legend('Airfoil', N1_legend, 'Interpreter', 'latex');
grid on;
axis equal;

subplot(3,1,2);
Cp1_title = (sprintf('NACA 0012 $C_p$ vs. Published Data for %d panels'
    , N_1));
hold on;
title(Cp1_title, 'Interpreter', 'latex');
xlabel('$x/c$', 'Interpreter', 'latex');
ylabel('$C_p$', 'Interpreter', 'latex');
plot(xc(1:N_1/2), Cp_1(1:N_1/2), '-.om', 'LineWidth', 2);
plot(xread, Cpread, 'r', 'LineWidth', 2);
xlim([0 c]);
ylim([-2 1]);
legend('Source Panel Method', 'Mason Panel Method', 'Interpreter', '
    latex');
grid on;

subplot(3,1,3);
Cp1_title1 = (sprintf('NACA 0012 $C_p$ vs. Experimental Data for %d
    panels', N_1));
hold on;
title(Cp1_title1, 'Interpreter', 'latex');
xlabel('$x/c$', 'Interpreter', 'latex');
ylabel('$C_p$', 'Interpreter', 'latex');
plot(xc(1:N_1/2), Cp_1(1:N_1/2), '-.om', 'LineWidth', 2);
plot(xread2, Cpread2, 'b', 'LineWidth', 2);
xlim([0 c]);
ylim([-2 1]);
legend('Source Panel Method', 'Experimental Data', 'Interpreter', '
    latex');
grid on;

%Plots for N = 100
figure(2);
set(gcf, 'Position',  [400, 100, 800, 800]);
subplot(3,1,1);
plot(X, Y, 'k', 'LineWidth', 2);
hold on;
plot(X2, Y2, '-.or', 'LineWidth', 1.25);
title('NACA 0012 Geometry', 'Interpreter', 'latex');
xlabel('$x/c$', 'Interpreter', 'latex');
ylabel('$y/c$', 'Interpreter', 'latex');
N2_legend = (sprintf('%d Source Panels', N_2));
legend('Airfoil', N2_legend, 'Interpreter', 'latex');
grid on;
axis equal;
```

```
subplot(3,1,2);
Cp2_title = (sprintf('NACA 0012 $C_p$ vs. Published Data for %d panels'
    , N_2));
hold on;
title(Cp2_title, 'Interpreter', 'latex');
xlabel('$x/c$', 'Interpreter', 'latex');
ylabel('$C_p$', 'Interpreter', 'latex');
plot(xc2(1:N_2/2), Cp_2(1:N_2/2), '-.om', 'LineWidth', 2);
plot(xread, Cpread, 'r', 'LineWidth', 2);
xlim([0 c]);
ylim([-2 1]);
legend('Source Panel Method', 'Mason Panel Method', 'Interpreter', '
    latex');
grid on;

subplot(3,1,3);
Cp2_title1 = (sprintf('NACA 0012 $C_p$ vs. Experimental Data for %d
    panels', N_2));
hold on;
title(Cp2_title1, 'Interpreter', 'latex');
xlabel('$x/c$', 'Interpreter', 'latex');
ylabel('$C_p$', 'Interpreter', 'latex');
plot(xc2(1:N_2/2), Cp_2(1:N_2/2), '-.om', 'LineWidth', 2);
plot(xread2, Cpread2, 'b', 'LineWidth', 2);
xlim([0 c]);
ylim([-2 1]);
legend('Source Panel Method', 'Experimental Data', 'Interpreter', '
    latex');
grid on;

%Plots for N = 200
figure(3);
set(gcf, 'Position',  [400, 100, 800, 800]);
subplot(3,1,1);
plot(X, Y, 'k', 'LineWidth', 2);
hold on;
plot(X3, Y3, '-.or', 'LineWidth', 1.25);
title('NACA 0012 Geometry', 'Interpreter', 'latex');
xlabel('$x/c$', 'Interpreter', 'latex');
ylabel('$y/c$', 'Interpreter', 'latex');
N3_legend = (sprintf('%d Source Panels', N_3));
legend('Airfoil', N3_legend, 'Interpreter', 'latex');
grid on;
axis equal;

subplot(3,1,2);
Cp3_title = (sprintf('NACA 0012 $C_p$ vs. Published Data for %d panels'
    , N_3));
hold on;
title(Cp3_title, 'Interpreter', 'latex');
xlabel('$x/c$', 'Interpreter', 'latex');
ylabel('$C_p$', 'Interpreter', 'latex');
plot(xc3(1:N_3/2), Cp_3(1:N_3/2), '-.om', 'LineWidth', 2);
```

```
plot(xread, Cpread, 'r', 'LineWidth', 2);
xlim([0 c]);
ylim([-2 1]);
legend('Source Panel Method', 'Mason Panel Method', 'Interpreter', '
    latex');
grid on;

subplot(3,1,3);
Cp3_title1 = (sprintf('NACA 0012 $C_p$ vs. Experimental Data for %d
    panels', N_3));
hold on;
title(Cp3_title1, 'Interpreter', 'latex');
xlabel('$x/c$', 'Interpreter', 'latex');
ylabel('$C_p$', 'Interpreter', 'latex');
plot(xc3(1:N_3/2), Cp_3(1:N_3/2), '-.om', 'LineWidth', 2);
plot(xread2, Cpread2, 'b', 'LineWidth', 2);
xlim([0 c]);
ylim([-2 1]);
legend('Source Panel Method', 'Experimental Data', 'Interpreter', '
    latex');
grid on;

%Plots for N = 400
figure(4);
set(gcf, 'Position',  [400, 100, 800, 800]);
subplot(3,1,1);
plot(X, Y, 'k', 'LineWidth', 2);
hold on;
plot(X4, Y4, '-.or', 'LineWidth', 1.25);
title('NACA 0012 Geometry', 'Interpreter', 'latex');
xlabel('$x/c$', 'Interpreter', 'latex');
ylabel('$y/c$', 'Interpreter', 'latex');
N4_legend = (sprintf('%d Source Panels', N_4));
legend('Airfoil', N4_legend, 'Interpreter', 'latex');
grid on;
axis equal;

subplot(3,1,2);
Cp4_title = (sprintf('NACA 0012 $C_p$ vs. Published Data for %d panels'
    , N_4));
hold on;
title(Cp4_title, 'Interpreter', 'latex');
xlabel('$x/c$', 'Interpreter', 'latex');
ylabel('$C_p$', 'Interpreter', 'latex');
plot(xc4(1:N_4/2), Cp_4(1:N_4/2), '-.om', 'LineWidth', 2);
plot(xread, Cpread, 'r', 'LineWidth', 2);
xlim([0 c]);
ylim([-2 1]);
legend('Source Panel Method', 'Mason Panel Method', 'Interpreter', '
    latex');
grid on;

subplot(3,1,3);
Cp4_title1 = (sprintf('NACA 0012 $C_p$ vs. Experimental Data for %d
```

```
    panels', N_4));
hold on;
title(Cp4_title1, 'Interpreter', 'latex');
xlabel('$x/c$', 'Interpreter', 'latex');
ylabel('$C_p$', 'Interpreter', 'latex');
plot(xc4(1:N_4/2), Cp_4(1:N_4/2), '-.om', 'LineWidth', 2);
plot(xread2, Cpread2, 'b', 'LineWidth', 2);
xlim([0 c]);
ylim([-2 1]);
legend('Source Panel Method', 'Experimental Data', 'Interpreter', '
    latex');
grid on;

%%NOTE: Also, although I am aware that it says to declare functions in
%%separate MATLAB files, I find that to be too clunky. When submitting
    the
%%final product, it becomes annoying submitting all of the separate
    files
%%and for graders to download them all.
%==============================Functions
    ====================================
function [xA, yA] = plotAirfoil()
%This function computes the coordinates of a NACA XXXX airfoil with
    input
%parameters N, which is the number of panels, and topStatus, which
    tells
%the function if the points are on the top or bottom surface (1 for top
     and
%2 for bottom). Function returns an array of the x and y coordinates.
t = 0.12;                                   %NACA 00XX, t is XX/100
c = 1;                                      %chord length
N = 1000;                                   %Number of points
xA_upper = linspace(0, c, (N/2)+1);         %precalculated evenly spaced x
    values from [0, c]
xA_lower = linspace(c, 0, N/2);             %precalculated evenly spaced x
    values from [c, 0]
xA = zeros(1,N+1);                          %array to store x values
yA = zeros(1,N+1);                          %array to store y values

for k=1:N+1
    if k <= N/2 + 1
        xA(k) = xA_upper(k);
        yA(k) = t/(0.2) * c * (0.2969 * sqrt(xA(k) / c) - 0.1260 * (xA(
            k)/c) - 0.3516 * (xA(k)/c)^2 + 0.2843 * (xA(k)/c)^3 - 0.1015
             * (xA(k)/c)^4);
    else
        xA(k) = xA_lower(k-(N/2) - 1);
        yA(k) = -t/(0.2) * c * (0.2969 * sqrt(xA(k) / c) - 0.1260 * (xA
            (k)/c) - 0.3516 * (xA(k)/c)^2 + 0.2843 * (xA(k)/c)^3 -
            0.1015 * (xA(k)/c)^4);
    end
end

end
```

```
function [X, Y] = panels(N)
%This function computes the coordinates of a NACA XXXX airfoil with
    input
%parameters N, which is the number of panels, and topStatus, which
    tells
%the function if the points are on the top or bottom surface (1 for top
     and
%2 for bottom). Function returns an array of the x and y coordinates.
t = 0.12;                                   %NACA 00XX, t is XX/100
c = 1;                                      %chord length
xA_upper = linspace(0, c, (N/2)+1);         %precalculated evenly spaced x
    values from [0, c]
xA_lower = linspace(c, 0, N/2);             %precalculated evenly spaced x
    values from [c, 0]
X = zeros(1,N+1);                           %array to store x values
Y = zeros(1,N+1);                           %array to store y values

for k=1:N+1
    if k <= N/2 + 1
        X(k) = xA_upper(k);
        Y(k) = t/(0.2) * c * (0.2969 * sqrt(X(k) / c) - 0.1260 * (X(k)/
            c) - 0.3516 * (X(k)/c)^2 + 0.2843 * (X(k)/c)^3 - 0.1015 * (X
            (k)/c)^4);
    else
        X(k) = xA_lower(k-(N/2) - 1);
        Y(k) = -t/(0.2) * c * (0.2969 * sqrt(X(k) / c) - 0.1260 * (X(k)
            /c) - 0.3516 * (X(k)/c)^2 + 0.2843 * (X(k)/c)^3 - 0.1015 * (
            X(k)/c)^4);
    end
end
end

function [xc, yc] = controlPoint(N, x, y)
%Calculates the control points of each panel from the x and y arrays.
%Takes in inputs N (number of panels), x and y (coordinates of the
%endpoints of the panels)

for k = 1:N
    xc(k) = (x(k) + x(k+1)) / 2;
    yc(k) = (y(k) + y(k+1)) / 2;
end

end

function Iij = integral(xi, yi, Xj, Yj, Xjp1, Yjp1, phi_i, phi_j)
%Calculate the ij integral component using the integral given by Eq.
    3.162
%and 3.163.
A = -(xi - Xj) * cos(phi_j) - (yi - Yj) * sin(phi_j);
B = (xi - Xj)^2 + (yi - Yj)^2;
C = sin(phi_i - phi_j);
D = (yi - Yj) * cos(phi_i) - (xi - Xj) * sin(phi_i);
E = sqrt(B - A^2);
```

```
Sj = sqrt((Xjp1 - Xj)^2 + (Yjp1 - Yj)^2);

Iij = (C / 2) * log((Sj^2 + 2*A*Sj + B) / B) + (D - A*C)/E * (atan2(Sj
    + A, E) - atan2(A, E));

end

function Jij = integral_j(xi, yi, Xj, Yj, Xjp1, Yjp1, phi_i, phi_j)
%Calculate the integral over the jth panel given by equation 3.165
A = -(xi - Xj) * cos(phi_j) - (yi - Yj) * sin(phi_j);
B = (xi - Xj)^2 + (yi - Yj)^2;
C = sin(phi_i - phi_j);
D = (yi - Yj) * cos(phi_i) - (xi - Xj) * sin(phi_i);
E = sqrt(B - A^2);
Sj = sqrt((Xjp1 - Xj)^2 + (Yjp1 - Yj)^2);

Jij = (D - A*C)/(2*E) * log((Sj^2 + 2*A*Sj + B) / B) - C * (atan2(Sj +
    A, E) - atan2(A, E));

end

function phi = phi_calc(X, Y, Xp1, Yp1)
%This function calculates phi (the angle of the panel with respect to
    the
%horizontal). Has four inputs, the start points of the panel and the
    end
%points of the panel.
delta_X = Xp1 - X;
delta_Y = Yp1 - Y;

phi = atan2(delta_Y, delta_X);

end

function Cpi = Cp(Vi, V_infty, N)
%This function calculate the pressure coefficient Cp with two inputs:
    the
%tangential velocity component, Vi, and the freestream velocity,
    V_infty.
for i=1:N
    Cpi(i) = 1 - (Vi(i) / V_infty)^2;
end

end

function sourceSum(lambda, sj, N)
%lambda and sj are arrays that store the source strength of each panel
    and
%length of each panel. It has no outputs and prints the sum of the
    source
%strength multiplied by the panel length.
product = zeros(1, N);

for i=1:N
```

```
    product(i) = lambda(i) * sj(i);
end

zeroCheck = sum(product, 'all');

fprintf('Sum of all sources/sinks for %d panels: %f\n', N, zeroCheck);

end

function Vi = tangentialVelocity(A_t, lambda, B_t, N)
%Calculates the tangential velocity of each panel, where B_t is defined
    by
%B_t = V_infty * sin(beta_i). The first three parameters are arrays.
Vi = zeros(N, 1);

for i = 1:N
    Vi(i) = Vi(i) + B_t(i);
    for j = 1:N
        Vi(i) = Vi(i) + A_t(i, j) * lambda(j);
    end
end
end
```