Two Element Linear Strength Vortex Panel Method

A Senior Project

presented to

the Faculty of the Aerospace Engineering Department

California Polytechnic State University, San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Science in Aerospace Engineering

by

Clifton A. Cox

March 2011

Linear Strength Vortex Panel Method for a Two Element Airfoil

Clifton Cox[1]

California Polytechnic State University, Aerospace Engineering Department, San Luis Obispo, CA, 93407

A linear strength vortex panel method was developed to predict the C_p and C_l for a lifting two element airfoil. The linear strength vortex panel method was first validated against thin airfoil theory and experimental data for a single NACA 2412 airfoil. At 2 degrees angle of attack, the linear strength vortex panel method predicted a C_l of about 0.49. Experimental data and thin airfoil theory gave C_l estimations of 0.45 and 0.22 respectively. The Matlab code was then modified to accept a two element airfoil. The two key modifications were the separation of the two different sets of wing element panels and the subsequent addition of a second Kutta condition. The linear strength vortex panel method was then used to determine the C_l and C_p distribution of a two element wing. The two element wing of study was the rear wing airfoil used on the 2008 Formula SAE car. Using a reference length of 1.43 and an angle of attack of 2 degrees, the panel method predicted a C_l of 3.98. Improved results can be obtained by using more panels or better geometry resolution around the leading edge and the gap between the two wing elements.

Nomenclature

C_l	=	lift coefficient
C_p	=	pressure coefficient
M	=	number of panels
N	=	number of equations to solve
a	=	influence coefficient
c	=	reference length
d_l	=	panel length
dx	=	distance between points in the x direction
dy	=	distance between points in the y direction
i	=	panel endpoint index
j	=	collocation point index
r	=	distance between panel edge and collocation point
u	=	x direction velocity in panel coordinates
w	=	y direction velocity in panel coordinates
α	=	angle of attack (main element, in degrees)
γ	=	vortex strength

[1]Student, Aerospace Engineering Department, 1 Grand Ave., San Luis Obispo 93407

δ	=	angle between flap and 0° main element (in degrees)
θ	=	panel orientation angle
subscripts		
1	=	first panel point coordinate
2	=	second panel point coordinate
main	=	values corresponding to the main element
flap	=	values corresponding to the flap
p	=	collocation point

I. Introduction

This paper presents the formulation of a two element airfoil panel method. This project was inspired by the development of an aerodynamics package for the 2008 Formula SAE car. During preliminary design, CFD methods were used to obtain lift, drag and pressure values. Significant amounts of time were spent up front to develop these complex CFD models. As a result, less time was available for manufacturing and testing. Another approach would have been to use simpler, faster models to obtain preliminary performance estimates, leaving more time for the manufacturing and testing of the design. Using a panel method could have provided quick performance estimates without the need for a complex model. This paper discusses the formulation of a panel method capable of analyzing a two element airfoil, as was used on the 2008 Formula SAE car.

Background

There are numerous ways to obtain lift and pressure estimates for a given airfoil. A rough estimate can be obtained from simple thin airfoil theory. However, this method is extremely general and loses accuracy as the airfoil thickness increases. Panel methods split the airfoil into separate panels to obtain lift and pressure estimates. Panel methods provide increased accuracy over thin airfoil theory and can be applied to a variety of airfoil configurations. CFD methods provide even more accuracy, but are more computationally expensive and complex than panel methods. For preliminary estimates, panel methods are both quick and accurate; thus the formulation of a two element airfoil panel method is worthwhile.

Panel methods break up an airfoil geometry into "panels" and then solve for the flow around the panels. There are many different panel method variations and each variation has its own strengths and weaknesses. Panel methods have two key features that distinguish themselves from each other: the formulation of the boundary conditions and the type of singularity element used to describe the flow field around the airfoil. The Neumann boundary condition states the normal component of the flow near the airfoil must be zero. On the other hand, the Dirichlet boundary condition sets the flow potential to be constant at the boundary. The second key feature is the type of singularity element. A source, doublet or vortex element can be used, but the panel method presented here uses the vortex element since it can model both lift and pressure.

Many panel method codes have already been developed. Depending on the airfoil geometry and desired complexity, previously developed codes can be found. Katz and Plotkin have many simple Fortran codes available for single element airfoils[1]. More complex 3D panel methods are also available for the modeling of an entire aircraft. This paper focuses on the development of a panel method that runs in the Matlab environment and can handle a two element airfoil. The key differences between a two element airfoil panel method and a single element airfoil panel method are the separation of the two geometries (so no panel has endpoints on two different elements) and a second Kutta condition (since the two distinct geometries provide a second unknown to solve for).

The first objective of this project is to write a Matlab script to run the Linear Strength Vortex type of panel method and then validate it against experimental data for a single element NACA 2412 airfoil. The second objective is to adapt the Linear Strength Vortex panel method so that it models a two element airfoil; in this case, the two element airfoil used for the 2008 Formula SAE rear wing is analyzed.

II. Analysis

The developed Matlab script has four key components. The geometry component takes a set of points describing the geometry of the airfoil and breaks it down into panels. The influence coefficients part of the Matlab script builds up a matrix relating the influence one panel has on every other panel. The solver part solves the large system of equations. Finally, post processing is done to obtain a lift coefficient and a pressure distribution.

The basic geometry is read in from a points file describing the "banana" airfoil. Scaling and positioning of the flap relative to the main element gives the desired shape of the two element airfoil. Figure 1 shows the two element airfoil geometry at zero angle of attack. The angle of attack is specified as the angle of the main element chord line with the free stream flow. The angle the flap makes with the main element is specified as delta. The points are then split into panels, with each pair of subsequent points as a new panel. The two remaining geometry values needed are the values for theta, the angle specifying the orientation of the panel, and the collocation point, or the point in the middle of the panel.

$$dx = x2 - x1 \tag{1.}$$

$$dy = y2 - y1 \tag{2.}$$

$$\theta = atan2(dy, dx) \tag{3.}$$

The collocation point is obtained by taking half the difference between the two points and adding it to the first point. Figure 2 gives a visual schematic of the panel geometry. Specifying the geometry becomes difficult with two elements since the panels specifying the main element have to be separated from the panels specifying the flap. Making sure there are no panels using one endpoint on the main element and the second endpoint on the flap is crucial to a working two element panel method. The Matlab code presented in this projects keeps both panel geometry endpoint matrices separate.

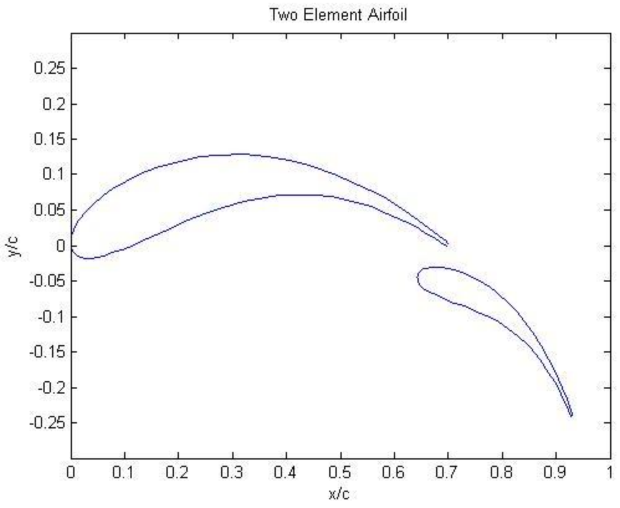

Figure 1: Two element airfoil with a main element at zero angle of attack.

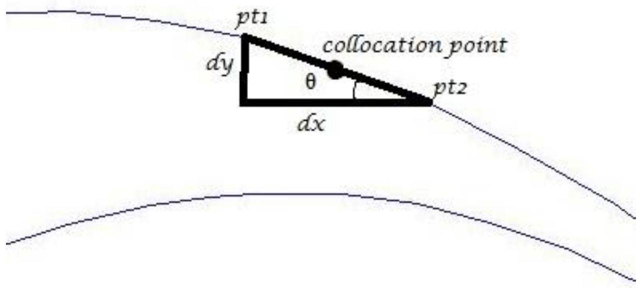

Figure 2: Panel geometry specification.

The influence coefficient matrix is built using a double for loop in Matlab, finding the influence every panel has on every other panel. The singularity element of choice was the linear strength vortex distribution. Since its a linear strength vortex distribution, the strength of the vortex, γ, is different on each edge of the panel. Equations 4 and 5 give the inducted velocity at any collocation point due to a linear strength panel[1].

$$u_p = \frac{Y}{2\pi}\left(\frac{\gamma_2-\gamma_1}{X_2-X_1}\right) ln\frac{r_2}{r_1} + \frac{\gamma_1(X_2-X_1)+(\gamma_2-\gamma_1)(X-X_1)}{2\pi(X_2-X_1)}(\theta_2-\theta_1) \qquad 4.$$

$$w_p = -\frac{\gamma_1(X_2-X_1)+(\gamma_2-\gamma_1)(X-X_1)}{2\pi(X_2-X_1)} ln\frac{r_1}{r_2} + \frac{Y}{2\pi}\left(\frac{\gamma_2-\gamma_1}{X_2-X_1}\right)\left(\frac{(X_2-X_1)}{Y} + (\theta_2-\theta_1)\right) \qquad 5.$$

In this equation, the unknowns are the induced velocities and the vortex strengths at each panel endpoint. After rearranging the equations, the induced velocity at a collocation point due to one vortex strength can be expressed in terms of this vortex strength. Equations are generating using the zero induced normal flow boundary condition. The influence coefficients become:

$$a_{ij} = (u,w)_{i,j} \cdot n_i \qquad 6.$$

Finding the influence coefficients for every combination yields a system of equations, in the form Ax=B, where x is a vector of unknown vortex strengths and B is the normal free stream flow. Only using these equations will produce one extra unknown, which is dealt with in the form of the Kutta condition, which states the flow leaving the trailing edge must be parallel at both the upper and lower endpoints. This methodology is standard for many panel methods using the Neumann boundary condition. With the addition of a two element airfoil, the influence of panels on the flap on collocation points on the main element are just as important as the influence of panels on the same airfoil element. Also, there are two Kutta condition equations, one for the trailing edge of each element. These differences make the two element airfoil panel method more complicated to implement in Matlab. Modification to the double for loop is required since there are two sets of lower trailing edge points and two sets of upper trailing edge points. If the geometry information is combined into one matrix, one of the first element's trailing edge points and one of the second element's trailing edge points will be in the middle of this matrix. Since trailing edge points have slightly modified equations, keeping track of which index refers to these points is crucial. The Matlab code presented here keeps track of the airfoil element corresponding to each index i and j throughout the double for loop, and adjusts equation usage accordingly. Also, the second Kutta condition refers to two points that are not the first and last points in the geometry specification with a two element airfoil panel method.

Solving the system of equations is extremely easy in Matlab. The panel method presented in this project uses the reduced row echelon function in Matlab to solve for all the unknown vortex element strengths.

The pressure distribution and lift coefficients are obtained from the panel velocities. The velocity at each panel is the summation of the induced velocity contributions of the other panels. These velocity contributions are obtained from the solved vortex strengths. The C_p distribution is obtained from solving equation 7 at each panel.

$$C_p = 1 - V^2 \qquad 7.$$

The coefficient of lift can be obtained by approximating the following integral[2]:

$$C_l = -\int C_p \hat{n} \cdot \hat{l} ds \qquad 8.$$

III. Results and Discussion

The linear strength vortex panel method was validated using a NACA 2412 airfoil. Thin airfoil theory predicted a C_l of 0.22 at 2 degrees angle of attack. The experimental data gave a C_l of approximately 0.45 at 2 degrees angle of attack[3]. The Matlab linear strength vortex panel method developed in this project gave a C_l of 0.49 at 2 degrees angle of attack. Figure 3 shows a C_l vs. angle of attack graph for all three methods. The experimental values closely resemble the panel method values during the linear region, indicating that the Matlab code is working correctly. The thin airfoil theory line is shifted down, but also shares the same slope as the two other methods. The difference in thin airfoil theory can be attributed to the thickness of the NACA 2412 airfoil.

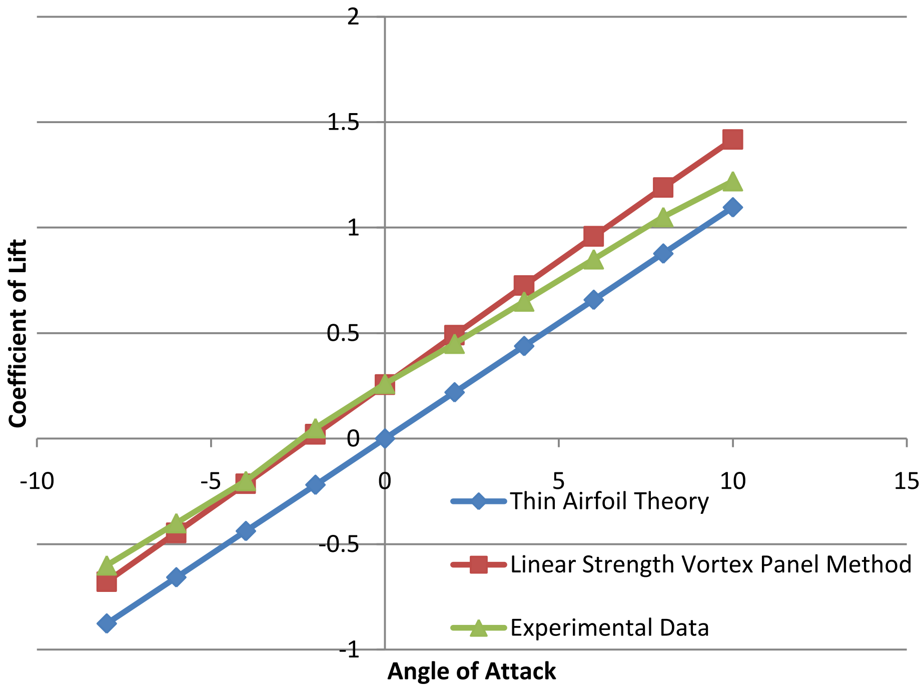

Figure 3: Angle of Attack vs. Lift Coefficient for a NACA 2412 Airfoil

The C_p calculation in the Matlab panel method code was validated against C_p data from a Mathematica code based off a NACA 4412 airfoil[2]. The chart below shows the C_p distribution of a NACA 4412 at 10 degrees angle of attack.

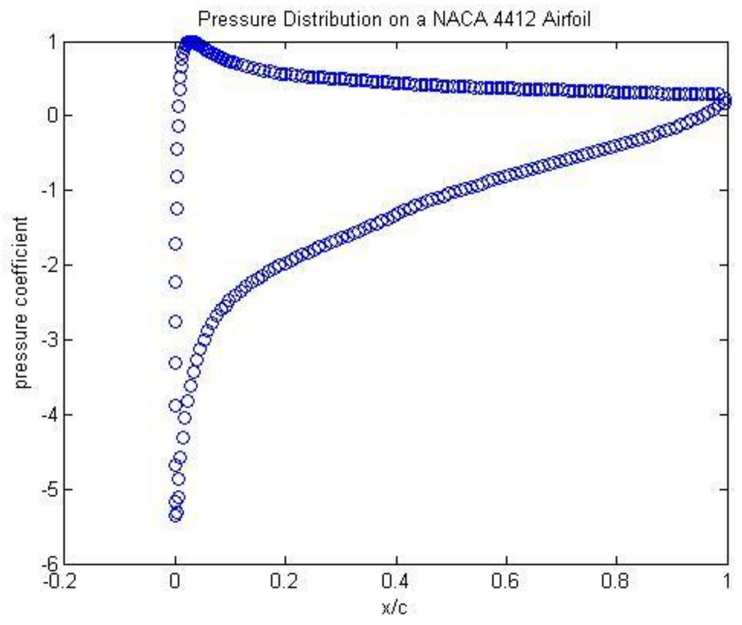

Figure 4: Pressure distribution for a NACA 4412 Airfoil at 10 degrees angle of attack.

The C_p plot shows the expected shape, confirming that both C_p and C_l are calculated correctly in the Matlab panel method presented here.

Now that the basic linear strength vortex panel method is validated, the second element to the airfoil is added. At 2 degrees angle of attack, the predicted C_l for the 2008 Formula SAE rear wing airfoil configuration is 3.98. This is based off a reference length extending from the leading edge of the main element to the trailing edge of the flap element. This lift coefficient is drastically higher than the single element coefficient of lift, which is expected. The addition of a second element to the airfoil configuration greatly increases the lift. Figure 5 shows the pressure distribution for the two element airfoil.

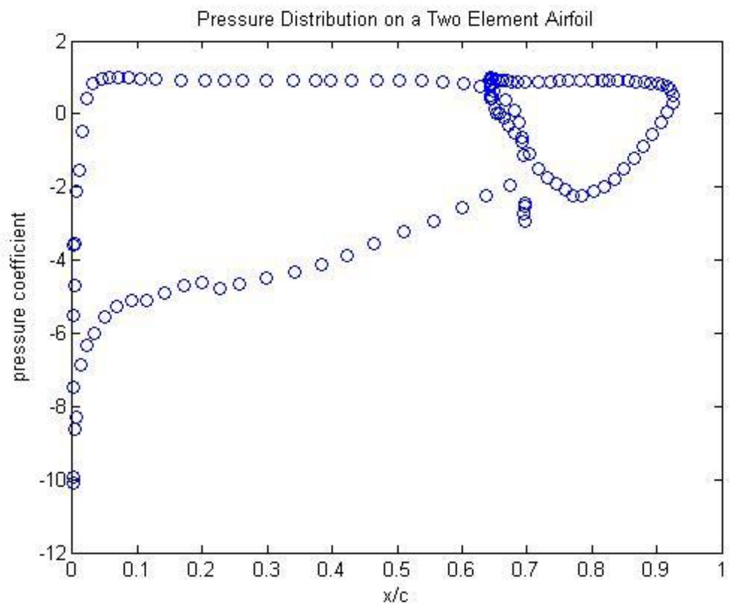

Figure 5: Pressure distribution for the two element Formula SAE rear wing at 2 degrees angle of attack.

This pressure distribution chart shows the expected trend. There appears to be several outliers in the region near the trailing edge of the main element. This is believed to be due to poor resolution in the trailing edge points distribution. Improved resolution on the trailing edge would shift those outliers up. It should also be noted that the rear wing in practice is oriented upside down, causing the lift vector to act in the downward direction, creating downforce instead of lift. In this case, the magnitude of the lift values will be the same, but in the opposite direction.

From the above results, the Matlab linear strength vortex panel method works for both single and dual element airfoils. The addition of a second element into the code provides a quick way to calculate C_l for a range of angle of attacks on two element airfoils. Also, the code only took a couple seconds to run, which is far shorter than a CFD solution. One key observation was made regarding solution accuracy. Accurate results are highly depended on the quality of the airfoil geometry provided, especially in the trailing edge and leading edge regions. Originally, an equal distribution of points along the airfoil was used as the geometry file. The results were alright, but a geometry file with a higher resolution of points near the trailing and leading edge yielded much better results. As stated previously, improving the quality of the points file used for the two element airfoil would smooth out the pressure distribution. The difference in smoothness is clearly shown in figure 4 vs. figure 5.

IV. Conclusion

Overall, the two element airfoil linear strength vortex panel method Matlab code was a success. The code provided quick, accurate results for the two element 2008 Formula SAE rear wing, which could have saved time during the preliminary design phase. With Matlab becoming more popular among engineering students and professionals, using Matlab as the coding environment is practical and useful for future work.

The key source of error is the geometry resolution. As with most numerical solutions, increasing the resolution increases the accuracy of the results. For this Matlab panel method in particular, it was found that the leading edge and the trailing edge were the important areas needing resolution. A high resolution at the leading edge is needed to sufficiently resolve the leading edge suction peak. The trailing edge resolution is important not only to resolve the Kutta condition, but also to model the interaction between the first and the second airfoil in the two element airfoil configuration.

This code could easily be adapted to any two element wing. The only changes would be to the geometry points files and the hardcoded distinction between the two elements. Future work could also include expansion to a 3 element wing (or higher!). Once again, in addition to adding additional Kutta conditions, separating the geometry and keeping the influence coefficients clear and organized is paramount.

References

1. Katz, J., Plotkin, A., *Low-Speed Aerodynamics*, 2nd ed., Cambridge University Press, 2001.

2. Fearn, Richard L., "Airfoil Aerodynamics Using Panel Methods," *The Mathematica Journal*, Volume 10, Issue 4, 2008.

3. Anderson, John D., *Fundamentals of Aerodynamics*, McGraw-Hill, New York, NY, 2007.

5. Matlab, Software Package, Ver. 2007b, MathWorks, Natick, MA, 2007.

Appendix

```
% Clif Cox
% 2011

% This script uses a linear strength vortex panel method to find the lift
% and pressure coefficients on a two element wing.  The points are taken
% from 2008 formula SAE rear wing. This file generates the points for the
% formula sae wing using the banana airfoil coordinates for both the main
% element and the flap.
clc
clear all
close all
% points file goes naturally from leading edge clockwise.  We want
% clockwise, lower trailing edge.
alpha = 2; %alpha is defined as angle of attack from main element chord line.
AL = alpha / 57.2958; % get into radians
ref_length = 1.43; % distance from leading edge of main element to trailing
                   % edge of flap

load banana_pts_refined % points file, unit length
x_main = F1_banana_hotwire(:,1);
y_main = F1_banana_hotwire(:,2);
x_flap = F1_banana_hotwire(:,1);
y_flap = F1_banana_hotwire(:,2);

% scale the flap
x_flap = 0.5.*x_flap;
y_flap = 0.5.*y_flap;

% rotate the flap angle delta 40 degres down
delta = 35;
x1 = zeros(size(x_flap));
y1 = zeros(size(y_flap));
for i = 1:length(x_flap)
    r       = (  x_flap(i)^2 + y_flap(i)^2  )^0.5;
    thetatr = atan2(y_flap(i), x_flap(i));
    theta   = thetatr - delta*pi/180;
    x1(i)  = r*cos(theta);
    y1(i)  = r*sin(theta);
end
x_flap = x1;
y_flap = y1;

% mode leading edge of flap to the right spot
x_flap = x_flap + .92;
y_flap = y_flap - .06;

% get rid of outlier points in file to improve point distribution
x_temp = x_main;
y_temp = y_main;
clear x_main y_main

for i = 1:30
    x_main(i)  = x_temp(i);
    y_main(i)  = y_temp(i);
end
for i = 32:69
    x_main(i-1)  = x_temp(i);
    y_main(i-1)  = y_temp(i);
end

x_temp = x_flap;
```

```
y_temp = y_flap;
clear x_flap y_flap

del = [28 29 30 31 32 34 35 36];
for i = 1:27
    x_flap(i) = x_temp(i);
    y_flap(i) = y_temp(i);
end
x_flap(28) = x_temp(33);
y_flap(28) = y_temp(33);
for i = 37:69
    x_flap(i-8) = x_temp(i);
    y_flap(i-8) = y_temp(i);
end

x_flap = x_flap';
x_main = x_main';
y_flap = y_flap';
y_main = y_main';

%convert so point one is upper trailing edge
n = 1;
for i = 31:68
    EPT_main(n,1) = x_main(i);
    EPT_main(n,2) = y_main(i);
    n=n+1;
end
for i = 1:30
    EPT_main(n,1) = x_main(i);
    EPT_main(n,2) = y_main(i);
    n=n+1;
end
n=1;
for i = 28:61
    EPT_flap(n,1) = x_flap(i);
    EPT_flap(n,2) = y_flap(i);
    n=n+1;
end
for i = 1:30
    EPT_flap(n,1) = x_flap(i);
    EPT_flap(n,2) = y_flap(i);
    n=n+1;
end

% Establish panel endpoints for each separate element
for i = 1:(length(x_flap)-1)
    flap_PT1(i,1) = EPT_flap(i,1);
    flap_PT2(i,1) = EPT_flap(i+1,1);
    flap_PT1(i,2) = EPT_flap(i,2);
    flap_PT2(i,2) = EPT_flap(i+1,2);
end

for i = 1:(length(x_main)-1)
    main_PT1(i,1) = EPT_main(i,1);
    main_PT2(i,1) = EPT_main(i+1,1);
    main_PT1(i,2) = EPT_main(i,2);
    main_PT2(i,2) = EPT_main(i+1,2);
end

M_flap =length(x_flap) - 1; % number of flap panels
M_main = length(x_main)-1; % number of main element panels

M = M_flap+ M_main; %total number of panels
```

```
N = M+2;

% Find panel angles theta (panel orientation angle)
for i = 1:M_flap
    DY_flap = flap_PT2(i,2) - flap_PT1(i,2);
    DX_flap = flap_PT2(i,1) - flap_PT1(i,1);
    TH_flap(i) = atan2(DY_flap,DX_flap);
    DL_flap(i) = sqrt(DX_flap^2 + DY_flap^2);
end

for i = 1:M_main
    DY_main = main_PT2(i,2) - main_PT1(i,2);
    DX_main = main_PT2(i,1) - main_PT1(i,1);
    TH_main(i) = atan2(DY_main,DX_main);
    DL_main(i) = sqrt(DX_main^2 + DY_main^2);
end

% Establish Collocation Points
for i = 1:M_flap
    CO_flap(i,1) = (flap_PT2(i,1)-flap_PT1(i,1))/2 + flap_PT1(i,1);
    CO_flap(i,2) = (flap_PT2(i,2)-flap_PT1(i,2))/2 + flap_PT1(i,2);
end

for i = 1:M_main
    CO_main(i,1) = (main_PT2(i,1)-main_PT1(i,1))/2 + main_PT1(i,1);
    CO_main(i,2) = (main_PT2(i,2)-main_PT1(i,2))/2 + main_PT1(i,2);
end

% merge all into one set
% do flap then main
CO = [CO_flap;CO_main];
TH = [TH_flap';TH_main'];
DL = [DL_flap';DL_main'];

for i = 1:M
    % determine if we're on flap or main

    if i<=M_flap
        %we're dealing with a collocation point on the flap
     for j = 1:M+1
        if j <= M_flap
            %we're dealing with both collocation points on flap

             % we find the influence coefficient for a specific collocation
             % point, i, on each of the pannels, j. Then we move on to the next
             % collocation point.

             % Convert Collocation Point To Local Panel Coords
                XT = CO(i,1) - flap_PT1(j,1);
                YT = CO(i,2) - flap_PT1(j,2);
                X2T = flap_PT2(j,1) - flap_PT1(j,1);
                Y2T = flap_PT2(j,2) - flap_PT1(j,2);

                X = XT*cos(TH(j)) + YT*sin(TH(j)); % collocation point
                Y = -XT*sin(TH(j)) + YT*cos(TH(j)); %collocation point
                X1 = 0;
                Y1 = 0;
                X2 = X2T*cos(TH(j)) + Y2T*sin(TH(j));
                Y2 = 0;

             % Find the length of r1,r2,theta1 and theta2
                R1 = sqrt((X-X1)^2 + (Y-Y1)^2); %length from panel point 1 to
collocation point, in panel coords
```

```
                R2 = sqrt((X-X2)^2 + (Y-Y2)^2); %length from panel point 2 to
collocation point, in panel coords
                TH1 = atan2(Y-Y1,X-X1);
                TH2 = atan2(Y-Y2,X-X2);

                if i == j
                    Y = 0;
                    TH1 = 0;
                end
                % Compute velocity components as functions of Gamma1 and gamma2.
                % Velocity of panel j due to collocation point i
                if i == j
                    U1L = -0.5*(X-X2)/(X2);
                    U2L = 0.5*(X)/(X2);
                    W1L = -0.15916;
                    W2L = 0.15916;
                else
                    U1L = -(Y*log(R2/R1)+X*(TH2-TH1)-X2*(TH2-TH1))/(6.28319*X2);
                    U2L = (Y*log(R2/R1) + X*(TH2-TH1))/(6.28319*X2);
                    W1L = -((X2-Y*(TH2-TH1)) - X*log(R1/R2) +
X2*log(R1/R2))/(6.28319*X2);
                    W2L = ((X2 - Y*(TH2-TH1))-X*log(R1/R2))/(6.28319*X2);
                end

                % Transform the local velocities into global velocity functions
                U1 = U1L*cos(-TH(j)) + W1L*sin(-TH(j));
                U2 = U2L*cos(-TH(j)) + W2L*sin(-TH(j));
                W1 = -U1L*sin(-TH(j)) + W1L*cos(-TH(j));
                W2 = -U2L*sin(-TH(j)) + W2L*cos(-TH(j));

                % Compute the coefficients of gamma in the influence matrix.
                if j == (1 | (M_flap+1))
                    A(i,1) = -U1*sin(TH(i)) + W1*cos(TH(i));
                    HOLDA = -U2*sin(TH(i)) + W2*cos(TH(i));
                    B(i,1) = U1*cos(TH(i)) + W1*sin(TH(i));
                    HOLDB = U2*cos(TH(i)) + W2*sin(TH(i));
                elseif j== (M_flap)
                    A(i,M_flap) = -U1*sin(TH(i)) + W1*cos(TH(i)) + HOLDA;
                    A(i,M_flap+1) = -U2*sin(TH(i)) + W2*cos(TH(i));
                    B(i,M_flap) = U1*cos(TH(i)) + W1*sin(TH(i)) + HOLDB;
                    B(i,M_flap+1) = U2*cos(TH(i)) + W2*sin(TH(i));
                else
                    A(i,j) = -U1*sin(TH(i)) + W1*cos(TH(i)) + HOLDA;
                    HOLDA = -U2*sin(TH(i)) + W2*cos(TH(i));
                    B(i,j) = U1*cos(TH(i)) + W1*sin(TH(i)) + HOLDB;
                    HOLDB = U2*cos(TH(i)) + W2*sin(TH(i));
                end

        elseif j>(M_flap+1)
            % collocation point i is on flap, collocation point j is on
            % main

            % we find the influence coefficient for a specific collocation
            % point, i, on each of the pannels, j. Then we move on to the next
            % collocation point.

            % since there are two separate airfoils, point j=M_flap+1
            % should not be solved for (this is the 2nd edge of the last panel).
            % During the for loop, any j after
            % this should be j-1, to reference the correct pannel, except when
referring to matrix A.

            % Convert Collocation Point To Local Panel Coords
```

```
                XT = CO(i,1) - main_PT1(j-M_flap-1,1);
                YT = CO(i,2) - main_PT1(j-M_flap-1,2);
                X2T = main_PT2(j-M_flap-1,1) - main_PT1(j-M_flap-1,1);
                Y2T = main_PT2(j-M_flap-1,2) - main_PT1(j-M_flap-1,2);

                X = XT*cos(TH(j-1)) + YT*sin(TH(j-1)); % collocation point
                Y = -XT*sin(TH(j-1)) + YT*cos(TH(j-1)); %collocation point
                X1 = 0;
                Y1 = 0;
                X2 = X2T*cos(TH(j-1)) + Y2T*sin(TH(j-1));
                Y2 = 0;

             % Find the length of r1,r2,theta1 and theta2
                R1 = sqrt((X-X1)^2 + (Y-Y1)^2); %length from panel point 1 to
collocation point, in panel coords
                R2 = sqrt((X-X2)^2 + (Y-Y2)^2); %length from panel point 2 to
collocation point, in panel coords
                TH1 = atan2(Y-Y1,X-X1);
                TH2 = atan2(Y-Y2,X-X2);

                if i == j-1
                    Y = 0;
                    TH1 = 0;
                end
                % Compute velocity components as functions of Gamma1 and gamma2.
                % Velocity of panel j due to collocation point i
                if i == j-1
                    U1L = -0.5*(X-X2)/(X2);
                    U2L = 0.5*(X)/(X2);
                    W1L = -0.15916;
                    W2L = 0.15916;
                else
                    U1L = -(Y*log(R2/R1)+X*(TH2-TH1)-X2*(TH2-TH1))/(6.28319*X2);
                    U2L = (Y*log(R2/R1) + X*(TH2-TH1))/(6.28319*X2);
                    W1L = -((X2-Y*(TH2-TH1)) - X*log(R1/R2) +
X2*log(R1/R2))/(6.28319*X2);
                    W2L = ((X2 - Y*(TH2-TH1))-X*log(R1/R2))/(6.28319*X2);
                end

                % Transform the local velocities into global velocity functions
                U1 = U1L*cos(-TH(j-1)) + W1L*sin(-TH(j-1));
                U2 = U2L*cos(-TH(j-1)) + W2L*sin(-TH(j-1));
                W1 = -U1L*sin(-TH(j-1)) + W1L*cos(-TH(j-1));
                W2 = -U2L*sin(-TH(j-1)) + W2L*cos(-TH(j-1));

                % Compute the coefficients of gamma in the influence matrix.
                if j == ( (M_flap+2))
                    A(i,M_flap+2) = -U1*sin(TH(i)) + W1*cos(TH(i));
                    HOLDA = -U2*sin(TH(i)) + W2*cos(TH(i));
                    B(i,M_flap+2) = U1*cos(TH(i)) + W1*sin(TH(i));
                    HOLDB = U2*cos(TH(i)) + W2*sin(TH(i));
                elseif j== ( M+1)
                    A(i,M+1) = -U1*sin(TH(i)) + W1*cos(TH(i)) + HOLDA;
                    A(i,N) = -U2*sin(TH(i)) + W2*cos(TH(i));
                    B(i,M+1) = U1*cos(TH(i)) + W1*sin(TH(i)) + HOLDB;
                    B(i,N) = U2*cos(TH(i)) + W2*sin(TH(i));
                else
                    A(i,j) = -U1*sin(TH(i)) + W1*cos(TH(i)) + HOLDA;
                    HOLDA = -U2*sin(TH(i)) + W2*cos(TH(i));
                    B(i,j) = U1*cos(TH(i)) + W1*sin(TH(i)) + HOLDB;
                    HOLDB = U2*cos(TH(i)) + W2*sin(TH(i));
                end
        else
```

```
            end
         end
         A(i,N+1) = cos(AL)*sin(TH(i))-sin(AL)*cos(TH(i));

        else
            %we're dealing with a collocation point on the main
          for j = 1:M+1
              if j <= M_flap
                  % CO point i is on main, CO point j is on flap

                % we find the influence coefficient for a specific collocation
                % point, i, on each of the pannels, j. Then we move on to the next
                % collocation point.

                % Convert Collocation Point To Local Panel Coords
                    XT = CO(i,1) - flap_PT1(j,1);
                    YT = CO(i,2) - flap_PT1(j,2);
                    X2T = flap_PT2(j,1) - flap_PT1(j,1);
                    Y2T = flap_PT2(j,2) - flap_PT1(j,2);

                    X = XT*cos(TH(j)) + YT*sin(TH(j)); % collocation point
                    Y = -XT*sin(TH(j)) + YT*cos(TH(j)); %collocation point
                    X1 = 0;
                    Y1 = 0;
                    X2 = X2T*cos(TH(j)) + Y2T*sin(TH(j));
                    Y2 = 0;

                 % Find the length of r1,r2,theta1 and theta2
                    R1 = sqrt((X-X1)^2 + (Y-Y1)^2); %length from panel point 1 to
collocation point, in panel coords
                    R2 = sqrt((X-X2)^2 + (Y-Y2)^2); %length from panel point 2 to
collocation point, in panel coords
                    TH1 = atan2(Y-Y1,X-X1);
                    TH2 = atan2(Y-Y2,X-X2);

                    if i == j
                        Y = 0;
                        TH1 = 0;
                    end
                    % Compute velocity components as functions of Gamma1 and gamma2.
                    % Velocity of panel j due to collocation point i
                    if i == j
                        U1L = -0.5*(X-X2)/(X2);
                        U2L = 0.5*(X)/(X2);
                        W1L = -0.15916;
                        W2L = 0.15916;
                    else
                        U1L = -(Y*log(R2/R1)+X*(TH2-TH1)-X2*(TH2-TH1))/(6.28319*X2);
                        U2L = (Y*log(R2/R1) + X*(TH2-TH1))/(6.28319*X2);
                        W1L = -((X2-Y*(TH2-TH1)) - X*log(R1/R2) +
X2*log(R1/R2))/(6.28319*X2);
                        W2L = ((X2 - Y*(TH2-TH1))-X*log(R1/R2))/(6.28319*X2);
                    end

                    % Transform the local velocities into global velocity functions
                    U1 = U1L*cos(-TH(j)) + W1L*sin(-TH(j));
                    U2 = U2L*cos(-TH(j)) + W2L*sin(-TH(j));
                    W1 = -U1L*sin(-TH(j)) + W1L*cos(-TH(j));
                    W2 = -U2L*sin(-TH(j)) + W2L*cos(-TH(j));

                    % Compute the coefficients of gamma in the influence matrix.
                    if j == (1)
                        A(i,1) = -U1*sin(TH(i)) + W1*cos(TH(i));
                        HOLDA = -U2*sin(TH(i)) + W2*cos(TH(i));
                        B(i,1) = U1*cos(TH(i)) + W1*sin(TH(i));
                        HOLDB = U2*cos(TH(i)) + W2*sin(TH(i));
                    elseif j== (M_flap)
                        A(i,M_flap) = -U1*sin(TH(i)) + W1*cos(TH(i)) + HOLDA;
                        A(i,M_flap+1) = -U2*sin(TH(i)) + W2*cos(TH(i));
                        B(i,M_flap) = U1*cos(TH(i)) + W1*sin(TH(i)) + HOLDB;
                        B(i,M_flap+1) = U2*cos(TH(i)) + W2*sin(TH(i));
                    else
                        A(i,j) = -U1*sin(TH(i)) + W1*cos(TH(i)) + HOLDA;
                        HOLDA = -U2*sin(TH(i)) + W2*cos(TH(i));
                        B(i,j) = U1*cos(TH(i)) + W1*sin(TH(i)) + HOLDB;
                        HOLDB = U2*cos(TH(i)) + W2*sin(TH(i));
                    end
              elseif j>(M_flap+1)
                % collocation point i is on main, collocation point j is on
                % main

                % we find the influence coefficient for a specific collocation
                % point, i, on each of the pannels, j. Then we move on to the next
                % collocation point.

                % since there are two separate airfoils, point j=M_flap+1
                % should not be solved for (this is the 2nd edge of the last panel).
                % During the for loop, any j after
                % this should be j-1, to reference the correct pannel, except when
    referring to matrix A.

                % Convert Collocation Point To Local Panel Coords
                    XT = CO(i,1) - main_PT1(j-M_flap-1,1);
                    YT = CO(i,2) - main_PT1(j-M_flap-1,2);
                    X2T = main_PT2(j-M_flap-1,1) - main_PT1(j-M_flap-1,1);
                    Y2T = main_PT2(j-M_flap-1,2) - main_PT1(j-M_flap-1,2);

                    X = XT*cos(TH(j-1)) + YT*sin(TH(j-1)); % collocation point
                    Y = -XT*sin(TH(j-1)) + YT*cos(TH(j-1)); %collocation point
                    X1 = 0;
                    Y1 = 0;
                    X2 = X2T*cos(TH(j-1)) + Y2T*sin(TH(j-1));
                    Y2 = 0;

                 % Find the length of r1,r2,theta1 and theta2
                    R1 = sqrt((X-X1)^2 + (Y-Y1)^2); %length from panel point 1 to
    collocation point, in panel coords
                    R2 = sqrt((X-X2)^2 + (Y-Y2)^2); %length from panel point 2 to
    collocation point, in panel coords
                    TH1 = atan2(Y-Y1,X-X1);
                    TH2 = atan2(Y-Y2,X-X2);

                    if i == j-1
                        Y = 0;
                        TH1 = 0;
                    end
                    % Compute velocity components as functions of Gamma1 and gamma2.
                    % Velocity of panel j due to collocation point i
                    if i == j-1
                        U1L = -0.5*(X-X2)/(X2);
                        U2L = 0.5*(X)/(X2);
                        W1L = -0.15916;
                        W2L = 0.15916;
                    else
                        U1L = -(Y*log(R2/R1)+X*(TH2-TH1)-X2*(TH2-TH1))/(6.28319*X2);
                        U2L = (Y*log(R2/R1) + X*(TH2-TH1))/(6.28319*X2);
                        W1L = -((X2-Y*(TH2-TH1)) - X*log(R1/R2) +
    X2*log(R1/R2))/(6.28319*X2);
                        W2L = ((X2 - Y*(TH2-TH1))-X*log(R1/R2))/(6.28319*X2);
                    end

                    % Transform the local velocities into global velocity functions
                    U1 = U1L*cos(-TH(j-1)) + W1L*sin(-TH(j-1));
                    U2 = U2L*cos(-TH(j-1)) + W2L*sin(-TH(j-1));
                    W1 = -U1L*sin(-TH(j-1)) + W1L*cos(-TH(j-1));
                    W2 = -U2L*sin(-TH(j-1)) + W2L*cos(-TH(j-1));

                    % Compute the coefficients of gamma in the influence matrix.
                    if j == ( (M_flap+2))
                        A(i,M_flap+2) = -U1*sin(TH(i)) + W1*cos(TH(i));
                        HOLDA = -U2*sin(TH(i)) + W2*cos(TH(i));
                        B(i,M_flap+2) = U1*cos(TH(i)) + W1*sin(TH(i));
                        HOLDB = U2*cos(TH(i)) + W2*sin(TH(i));
                    elseif j== M+1
                        A(i,M+1) = -U1*sin(TH(i)) + W1*cos(TH(i)) + HOLDA;
                        A(i,N) = -U2*sin(TH(i)) + W2*cos(TH(i));
                        B(i,M+1) = U1*cos(TH(i)) + W1*sin(TH(i)) + HOLDB;
                        B(i,N) = U2*cos(TH(i)) + W2*sin(TH(i));
                    else
                        A(i,j) = -U1*sin(TH(i)) + W1*cos(TH(i)) + HOLDA;
                        HOLDA = -U2*sin(TH(i)) + W2*cos(TH(i));
                        B(i,j) = U1*cos(TH(i)) + W1*sin(TH(i)) + HOLDB;
                        HOLDB = U2*cos(TH(i)) + W2*sin(TH(i));
                    end
              else
              end
          end
          A(i,N+1) = cos(AL)*sin(TH(i))-sin(AL)*cos(TH(i));

        end
    end

       % Add both kutta conditions.  Be careful of where the ones are.
       % matrix columns M_flap+1 and M+2 are the last edges of the airfoil.

    %flap
    A(N-1,1) = 1;
    A(N-1,M_flap+1) = 1;
    %main
    A(N,M_flap+2) = 1;
    A(N,N) = 1;

    R = rref(A); % solve
    G = R(:,N+1);
    CL = 0;
    % calculate variables of interest
    for i = 1:M
        VEL = 0;
        for j = 1:N
            VEL = VEL + B(i,j)*G(j);
        end
        V = VEL + cos(AL)*cos(TH(i)) + sin(AL)*sin(TH(i));
        CP(i) = 1-V^2;
        CL = CL + -1.*CP(i)*(cos(AL)*cos(TH(i)) + sin(AL)*sin(TH(i)))*DL(i);
    end
    CP = CP';

    % report values of interest
    CL = CL/ref_length % varies based on ref. length definition
```

```
unitxm = EPT_main(:,1)./ref_length;
unitxf = EPT_flap(:,1)./ref_length;
unitym = EPT_main(:,2)./ref_length;
unityf = EPT_flap(:,2)./ref_length;
figure
plot(unitxm, unitym)
title('Two Element Airfoil')
hold on
plot(unitxf,unityf)
xlabel('x/c')
ylabel('y/c')
axis([0 1 -.3 .3])

figure
plot(CO(:,1)./ref_length,CP,'o')
title('Pressure Distribution on a Two Element Airfoil')
xlabel('x/c')
ylabel('pressure coefficient')
```