# Modeling Arbitrary NACA 4-Digit Airfoils with Kuethe Chow Vortex Panel Method

Patrick M. Kennedy
*Embry-Riddle Aeronautical University, Daytona Beach FL, USA*

**This technical report presents a comprehensive numerical analysis of the aerodynamic characteristics of NACA 4-digit airfoils utilizing the Kuethe Chow vortex panel method. The airfoil surface is discretized into a series of panels, and the vortex panel strengths are computed to satisfy the imposed boundary conditions. The pressure distribution, lift, drag, and pitching moment coefficients around the airfoil are thoroughly examined and compared with experimental data obtained from wind tunnel tests. The results demonstrate a high degree of agreement with the experimental data, particularly when comparing the rate of change of coefficients. The Kuethe Chow method proves to be a valuable alternative to Computational Fluid Dynamics (CFD) for predicting the aerodynamic performance of airfoils.**

## Nomenclature

| | |
|---|---|
| *a_Cl0* | *= Angle of attack of zero lift* |
| *a0* | *= Rate of change of Cl per AoA* |
| *AoA* | *= Angle of Attack* |
| *CFD* | *= Computation Fluid Dynamics* |
| *Cl_a0* | *= Lift coefficient of zero angle of attack* |
| *Cmac* | *= Coefficient of Moment about aerodynamic center* |
| *Cmc/4* | *= Coefficient of Moment of quarter chord* |
| *Cmle* | *= Coefficient of Moment about the leading edge* |
| *Cp* | *= Coefficient of Pressure* |
| *dCmc4/dAoA* | *= Rate of change of the coefficient of moment of quarter chord per AoA* |
| *LE* | *= Leading Edge* |
| *m* | *= Number of Panels* |
| *MCL* | *= Mean Camber Line* |
| *P&H* | *= Perkins and Hage* |
| *TE* | *= Trailing Edge* |
| *VPM* | *= Vortex Panel Method* |
| *x/c* | *= x location in % chord* |
| *Xac* | *= x location of the aerodynamic center* |
| *Xcp* | *= x location of the center of pressure* |
| *y/c* | *= y location in % chord* |

## Introduction

NACA airfoils have been widely used in the aerospace industry due to their superior aerodynamic performance. Various numerical methods have been developed to accurately predict their characteristics, including the source panel method. However, this method is limited in its lack of circulation, thus unable to predict lift. The Kuethe Chow vortex panel method addresses this limitation by considering vortex shedding and circulation around the airfoil. Moreover, its computation time is significantly shorter than that of CFD, making it a valuable alternative for practical engineering applications.

## I. Problem Statement

The Kuethe Chow vortex panel method represents an airfoil body as an infinitesimal summation of vortices forming a vortex sheet. This vortex sheet induces a circulation around the airfoil, which can be integrated to determine the lift produced on the airfoil via the Kutta-Joukowski theorem. This process is known as the Vortex Panel Method (VPM). VPM can be applied to either the upper and lower surfaces of an airfoil or to the mean camber line (MCL) in conjunction with thin airfoil theory.

The accuracy of VPM is evaluated in this paper. Results are compared to experimental data referenced in the Perkins and Hage airfoil tables.

## II. Program Structure

All code utilized for the analysis is implemented in MATLAB and referenced in the appendix.

Before defining the vortex, sheet and iterating across each panel, the geometry of the airfoil should be defined. The function “naca4digit” will accept a 4-character string and parse the digits into a struct containing all necessary parameters of the NACA 4-digit airfoil specified.

Once the geometry is defined, the program iterates over a list of angles of attack (AoA) values and executes the “vortex_panel_method” function for each airfoil orientation. Flow characteristics are computed within this function using VPM and returned as an output struct to the main scope.

The main scope iterates over all flow characteristics for each AoA and ultimately determines the aerodynamic center of the NACA 4-digit airfoil.

It is assumed the reader has sufficient knowledge in incompressible aerodynamics so not every equation and technique are stated in detail. The code prepared is shown in Appendix C and contains all equations and methodology necessary.

### a. naca4digit

The function “naca4digit” accepts 2 parameters: *digits* and *n*. *digits* will specify the NACA 4-digit airfoil to be modeled. This parameter does not accept modified NACA 4-digits and only standard 4 number definitions. *n* is the reference for how many panels are to be used in the NACA 4-digit model. *n* must be an even number, if odd the program will round down to the nearest even number.

The *digits* parameter is parsed to get the max camber, distance of max camber, and max thickness of the airfoil. Equations defined in the NACA Technical Report 824 were utilized to define function handles for the upper and lower surface of the airfoil.

The airfoil vertices are defined by assessing the upper and lower body coordinates over a cosine spacing from 0 to 180 degrees about the midchord of the airfoil. The algorithm will produce *n+1* vertices, which amounts to *n* number of panels. The leading edge (LE) and trailing edge (TE) are fixed at *(0,0)* and *(1,0)* respectively. An example of this cosine spacing is done for a 12-panel cut in *Figure 1*.

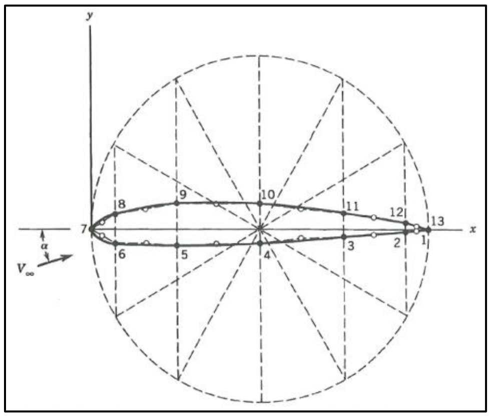


*Figure 1: Cosine spacing on x-axis for 12-panel model.*

### b. vortext_panel_method

The function "vortext_panel_method" accepts three parameters: *naca4*, *aoa_deg*, and *fig*. The following is a brief description of each parameter and their use case.

*naca4*: A struct that represents the airfoil to be simulated. The output of "naca4digits" is a struct that contains all members required of the *naca4* parameter. The necessary members of the parameter are a body definition in (*x/c, y/c)* coordinates, the number of panels, and the x intercept from the cosine spacing.

*aoa_deg*: A double representing the angle of attack to be evaluated in degrees. AoA is converted into radians immediately after the function is run. All results from the VPM are unique to the specific *naca4* and *aoa_deg* definitions.

*fig*: An int representing which figure to plot the coefficient of pressure (Cp) vs AoA curve. If *fig* is -1 or undefined the function will not plot the curve.

## III. VPM Implementation

Once all panels are modeled the Kuethe Chow VPM begins. The method revolves around the idea of no penetration boundary condition that states that the velocity normal to a control panel must be 0. This can be modeled using velocity potential Φ. Velocity potential follows super positions such that all vortex sheet's elements can be expressed in each panel's velocity potential.

This velocity potential can be expressed as a system of linear equations, containing geometric, uniform flow properties and vortex strength, which is effortless to solve using MATLAB. Since geometric properties and uniform flow are known, the vortex strength can be evaluated using a left matrix divide.

Once the vortex strength is known, velocity potential can be used to determine the velocity across each panel. Since this VPM only applies to incompressible, inviscid, attached flow, the velocity of the fluid above each panel is considered constant. The coefficient of pressure (Cp) in this case is only dependent on velocity, and thus, also constant per each panel.

The Cp distribution can now be integrated to determine the normal and axial coefficients acting on the airfoil. The normal and axial coefficients can be transformed into lift and drag coefficients using a rotation matrix about the specified AoA.

The Cp distribution is also used with geometric parameters to calculate the coefficient about the leading edge (Cmle). The Cmle acts from the center of pressure about the leading edge, thus this value is used to calculate the center of pressure (Xcp). Finally, the center of pressure, in pair with the coefficient of lift, can be used to calculate the coefficient of the moment about 25% of the chord (Cmc/4).

## IV. Airfoil Characteristics

Once all AoA dependent values are computed using the VPM, a thorough analysis on airfoil characteristics can be done.

### a. Lift and Moment over Angle of Attack

Determining the Cl vs AoA characteristics is an important aspect of airfoil design. Since Cl has been evaluated across multiple angles of attacks using the VPM, a relationship between the two can be developed.

For a standard airfoil, Cl vs AoA is expected to be linear with a slope of a0. The set of Cl vs AoA can be expressed in a form of y = mx+b by utilizing the MATLAB function polyfit, where a0 = m. In addition, the zero lift angle of attack can is also determined.

Along with Cl, Cmc/4 vs AoA is also typically linear. A new set of Cmc/4 vs AoA can be used with MATLAB's polyfit to read the change of Cmc/4 over AoA (dCmc4/dAoA).

### b. Aerodynamic Center

Finally, the aerodynamic center (Xac) can be computed. The following equation related Xac to the change of Cmc/4 and Cl over AoA.

$$Xac = 0.25 - \frac{\frac{dCmc/4}{dAoA}}{\frac{dCl}{dAoA}}$$

Once Xac is computed, the coefficient of moment about Xac can be computed using the following.

$$Cmac = -Cl\,(Xcp - Xac)$$

Cmac is considered constant with respect to AoA, though the equation provided shows a direct relationship to two functions of AoA, Cl and Xcp. Thus, when computing Cmac, a vector of values will be returned, a final polyfit is required to normalize the vector and identify error.

## V. Results

For analysis, the NACA 2412 airfoil was modeled at various number of panels and angle of attack.

To reduce error, a large range of panels used to determine flow characteristics were used across a range of angle of attack from -4 to 12 degrees. AoA > 12 degrees was not tested because the VPM does not account for high viscosity and detached flow, thus this would increase the number of outliers in the data.
Airfoil characteristics vary with panel count as shown in figure 2, averages of these values are shown in figure 3. In addition, plots representing the body, characteristics vs AoA, and Cmac vs Cl are shown in Appendix B – MATLAB Output.

| M | 12 | 48 | 120 | 2400 |
|---|---|---|---|---|
| ***A0*** | 0.077674 | 0.080965 | 0.082703 | 0.086241 |
| ***A_CL0*** | -2.46868 | -0.93242 | -0.99945 | -2.82219 |
| ***CL_A0*** | 0.191752 | 0.075493 | 0.082658 | 0.24339 |
| ***XAC*** | 0.55107 | 0.679178 | 0.713852 | 0.729732 |
| ***CMAC*** | 0.002515 | 0.029027 | 0.041356 | 0.082509 |

Figure 2: Tabulated NACA 2412 characteristics from multiple VPM test case.

| Variable | Average |
|---|---|
| ***a0*** | 0.081896 |
| ***a_Cl0*** | -1.80569 |
| ***Cl_a0*** | 0.148323 |
| ***Xac*** | 0.668458 |
| ***Cmac*** | 0.038852 |

Figure 3: Tabulated averages of NACA 2412 Airfoil Characteristics.

There is disagreement between a few of the trials. The Cl vs AoA slope, a0, is very consistent, the increase in a0 over an increase in number of panels can be explained by less of an uncertainty range the more panels are divided. It should be noted that a0 is measured in 1/deg, the rest of the tabulated results are unitless.

The AoA of zero lift, a_Cl0, and the coefficient of lift at zero AoA, Cl_a0, vary quite a bit, and in a non-linear pattern. a_Cl0 and Cl_a0 at 12 panels and 2400 panels agree which could indicate uncertainty for a medium number of panels. This could be because in a small set, such as m = 12, the nose and tail of the airfoil is not modeled, and for medium range, such as m = 48 or 120, the nose or tail are poorly modeled. Once 2400 panels are introduced the nose and tail can be accurately modeled and error is reduced.

The aerodynamic center, Xac, agrees with data from various panels. There is a slight increase in Xac as the number of panels increase, this can be explained by a higher fidelity as more refined panel models are introduced.

As a result of the increase in Xac, Cmac also increases for higher fidelity models. However, Cmac remains relatively stable for each individual trial as shown in Appendix B.3. Note the axis are scaled and variation of Cmac over Cl is slim.

By examining the center of pressure across AoA for all panel cases in Appendix B.2, it can be determined that for higher fidelity models (m > 12), Xcp is unstable at low angle of attacks. Since Xcp is derived from Cmle, at low angle of attacks Cl and Cmc/4 are very small which increases the uncertainty in determining Xcp.

Cl vs AoA is consistently linear as shown in Appendix B.2. There is very slight variation between Cl from VPM and from the polyfit line. Likewise for the moment about the quarter chord point Cmc/4.

## VI. Perkins and Hage Comparison

To understand the accuracy of this developed model a comparison to experimental data is necessary. Experimental results from Perkins and Hage (P&H) NACA Airfoil Section Data from Appendix A are used in analysis of this model. See figure 4 for percent differences between the averages of the VPM model and P&H experimental data.

| Variable | P&H Value | % difference |
|---|---|---|
| ***a0*** | 0.104 | 26.99072 |
| ***a_Cl0*** | -1.2 | 33.54332 |
| ***Cl_a0*** | NA | NA |
| ***Xac*** | 0.247 | 63.04929 |
| ***Cmac*** | -0.047 | 220.9727 |

Figure 4: Tabulated comparison of Perkins and Hage characteristics vs the derived VPM characteristics.

The lift curve slope, a0, has the smallest percent difference. From the finding in the results section, it should be noted that as the number of panels in increased, a0 approaches the P&H value of 0.104. Confidently, the software models the lift curve slope in accordance with experimental data.

The AoA of zero lift, a_Cl0 was previously labeled as non-linear about the number of panels. However, comparing the average of a_Cl0 to P&H values resulted in a lower percent difference than expected for a currently unpredictable data set. Improvement is needed on the calculation of a_Cl0.

The aerodynamic center has a larger percent error. Since Xac is a function of the rate of change of lift and Cmc/4 over AoA, and the Cmc/4 values are non-linear, the Xac experiences more error than ideal. The percent error of 63% can and should be improved on.

The coefficient of moment about the aerodynamic center has the largest percent difference of 220%. This is seemingly inaccurate, though when comparing the VPM results to the absolute value of the P&H Cmc/4, the percent difference becomes 21%. This could be explained by the error in the Xac, if the Xac error was slim, the pressure distribution applies the correct direction and magnitude, thus reducing the error in Cmac.

## VII. Conclusion

In conclusion, the VPM implementation is not recommended as a substitute for CFD or other modeling methods. The model demonstrates accuracy in determining the vortex elements but lacks in interpolating that data into airfoil characteristics. The percent difference between VPM values and P&H experimental values is large enough to revise methodology used in this code.

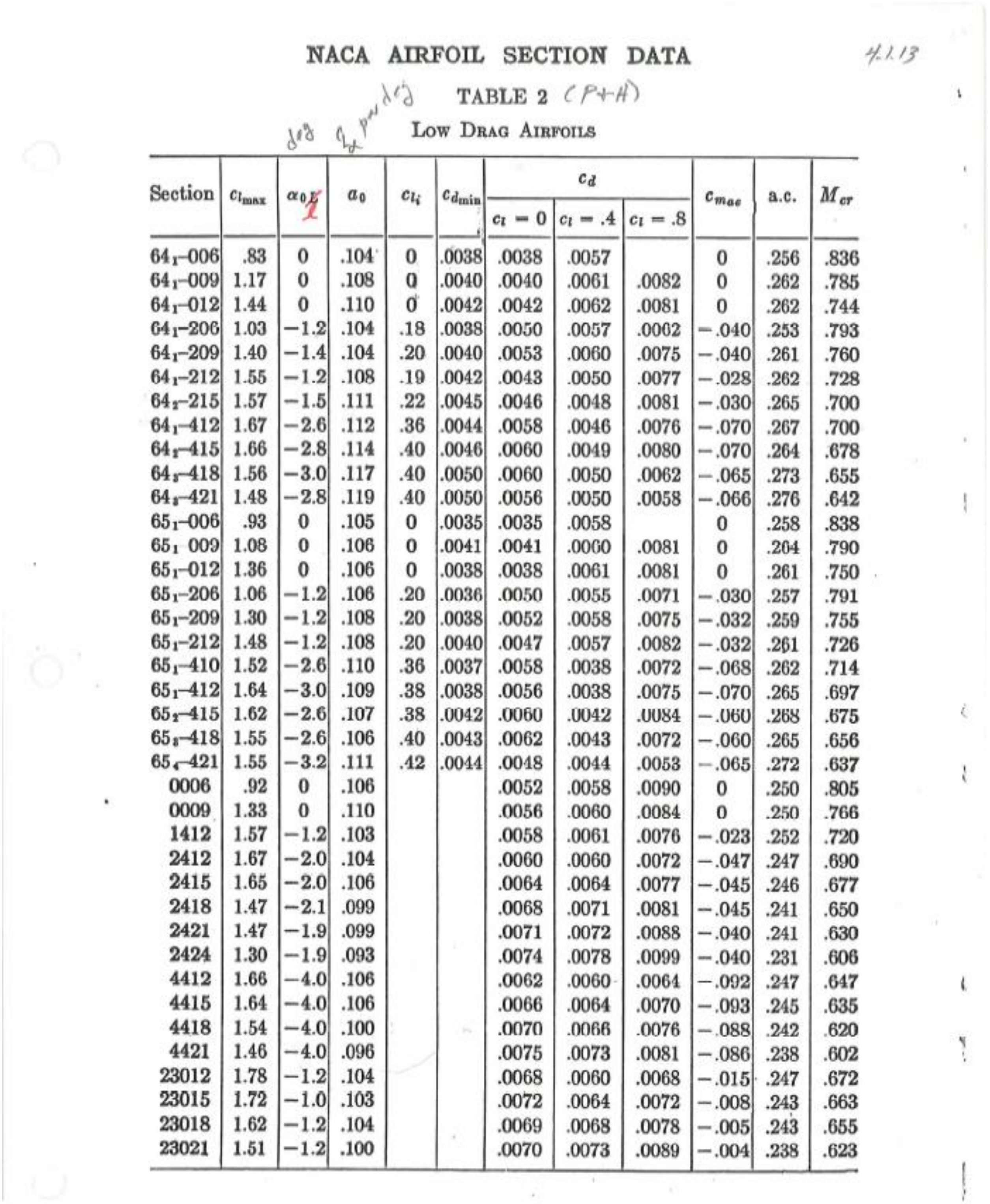
### a. Areas of Improvement

A large amount of error can be traced to poor calculation of Xcp. For future improvement, numerical methods of integrating across the Cp distribution should be implemented to determine the center of pressure with higher fidelity for a given airfoil at a given angle of attack. The correction factor of Xcp, specifically in low angle of attacks, could greatly improve the determination of Xac and thus Cmac.

In addition, if the calculation Xcp is improved and no longer a function of Cmle, the value of Cmle can increase in fidelity by using the pressure distribution and Xcp rather than numerical integration methods that could increase uncertainty.

Finally, correction factors for Xcp at low angles of attack would greatly decrease the error. Currently, the program will compute Xcp of negative values which is obviously incorrect and should be accounted for. On the case that Xcp is defined as negative, the program should re-construct the airfoil panels in hopes to model the LE and TE more accurately.

## VIII. Appendix A - Perkins and Hage NACA Airfoil Section Data



4.1.13

NACA AIRFOIL SECTION DATA

deg $c_{l_\alpha}$ per deg

TABLE 2 (P+H)

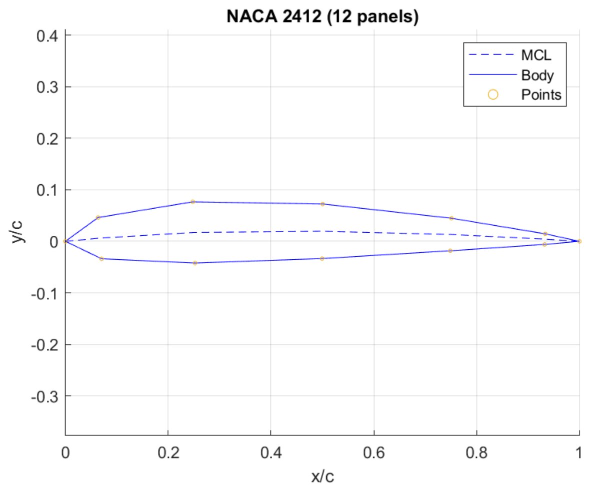
Low Drag Airfoils

| Section | $c_{l_{max}}$ | $\alpha_{0l}$ | $a_0$ | $c_{l_i}$ | $c_{d_{min}}$ | $c_d$ | | | $c_{m_{ac}}$ | a.c. | $M_{cr}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | $c_l = 0$ | $c_l = .4$ | $c_l = .8$ | | | |
| $64_1$–006 | .83 | 0 | .104 | 0 | .0038 | .0038 | .0057 | | 0 | .256 | .836 |
| $64_1$–009 | 1.17 | 0 | .108 | 0 | .0040 | .0040 | .0061 | .0082 | 0 | .262 | .785 |
| $64_1$–012 | 1.44 | 0 | .110 | 0 | .0042 | .0042 | .0062 | .0081 | 0 | .262 | .744 |
| $64_1$–206 | 1.03 | −1.2 | .104 | .18 | .0038 | .0050 | .0057 | .0062 | −.040 | .253 | .793 |
| $64_1$–209 | 1.40 | −1.4 | .104 | .20 | .0040 | .0053 | .0060 | .0075 | −.040 | .261 | .760 |
| $64_1$–212 | 1.55 | −1.2 | .108 | .19 | .0042 | .0043 | .0050 | .0077 | −.028 | .262 | .728 |
| $64_2$–215 | 1.57 | −1.5 | .111 | .22 | .0045 | .0046 | .0048 | .0081 | −.030 | .265 | .700 |
| $64_1$–412 | 1.67 | −2.6 | .112 | .36 | .0044 | .0058 | .0046 | .0076 | −.070 | .267 | .700 |
| $64_2$–415 | 1.66 | −2.8 | .114 | .40 | .0046 | .0060 | .0049 | .0080 | −.070 | .264 | .678 |
| $64_3$–418 | 1.56 | −3.0 | .117 | .40 | .0050 | .0060 | .0050 | .0062 | −.065 | .273 | .655 |
| $64_3$–421 | 1.48 | −2.8 | .119 | .40 | .0050 | .0056 | .0050 | .0058 | −.066 | .276 | .642 |
| $65_1$–006 | .93 | 0 | .105 | 0 | .0035 | .0035 | .0058 | | 0 | .258 | .838 |
| $65_1$ 009 | 1.08 | 0 | .106 | 0 | .0041 | .0041 | .0060 | .0081 | 0 | .264 | .790 |
| $65_1$–012 | 1.36 | 0 | .106 | 0 | .0038 | .0038 | .0061 | .0081 | 0 | .261 | .750 |
| $65_1$–206 | 1.06 | −1.2 | .106 | .20 | .0036 | .0050 | .0055 | .0071 | −.030 | .257 | .791 |
| $65_1$–209 | 1.30 | −1.2 | .108 | .20 | .0038 | .0052 | .0058 | .0075 | −.032 | .259 | .755 |
| $65_1$–212 | 1.48 | −1.2 | .108 | .20 | .0040 | .0047 | .0057 | .0082 | −.032 | .261 | .726 |
| $65_1$–410 | 1.52 | −2.6 | .110 | .36 | .0037 | .0058 | .0038 | .0072 | −.068 | .262 | .714 |
| $65_1$–412 | 1.64 | −3.0 | .109 | .38 | .0038 | .0056 | .0038 | .0075 | −.070 | .265 | .697 |
| $65_2$–415 | 1.62 | −2.6 | .107 | .38 | .0042 | .0060 | .0042 | .0084 | −.060 | .268 | .675 |
| $65_3$–418 | 1.55 | −2.6 | .106 | .40 | .0043 | .0062 | .0043 | .0072 | −.060 | .265 | .656 |
| $65_4$–421 | 1.55 | −3.2 | .111 | .42 | .0044 | .0048 | .0044 | .0053 | −.065 | .272 | .637 |
| 0006 | .92 | 0 | .106 | | | .0052 | .0058 | .0090 | 0 | .250 | .805 |
| 0009 | 1.33 | 0 | .110 | | | .0056 | .0060 | .0084 | 0 | .250 | .766 |
| 1412 | 1.57 | −1.2 | .103 | | | .0058 | .0061 | .0076 | −.023 | .252 | .720 |
| 2412 | 1.67 | −2.0 | .104 | | | .0060 | .0060 | .0072 | −.047 | .247 | .690 |
| 2415 | 1.65 | −2.0 | .106 | | | .0064 | .0064 | .0077 | −.045 | .246 | .677 |
| 2418 | 1.47 | −2.1 | .099 | | | .0068 | .0071 | .0081 | −.045 | .241 | .650 |
| 2421 | 1.47 | −1.9 | .099 | | | .0071 | .0072 | .0088 | −.040 | .241 | .630 |
| 2424 | 1.30 | −1.9 | .093 | | | .0074 | .0078 | .0099 | −.040 | .231 | .606 |
| 4412 | 1.66 | −4.0 | .106 | | | .0062 | .0060 | .0064 | −.092 | .247 | .647 |
| 4415 | 1.64 | −4.0 | .106 | | | .0066 | .0064 | .0070 | −.093 | .245 | .635 |
| 4418 | 1.54 | −4.0 | .100 | | | .0070 | .0066 | .0076 | −.088 | .242 | .620 |
| 4421 | 1.46 | −4.0 | .096 | | | .0075 | .0073 | .0081 | −.086 | .238 | .602 |
| 23012 | 1.78 | −1.2 | .104 | | | .0068 | .0060 | .0068 | −.015 | .247 | .672 |
| 23015 | 1.72 | −1.0 | .103 | | | .0072 | .0064 | .0072 | −.008 | .243 | .663 |
| 23018 | 1.62 | −1.2 | .104 | | | .0069 | .0068 | .0078 | −.005 | .243 | .655 |
| 23021 | 1.51 | −1.2 | .100 | | | .0070 | .0073 | .0089 | −.004 | .238 | .623 |

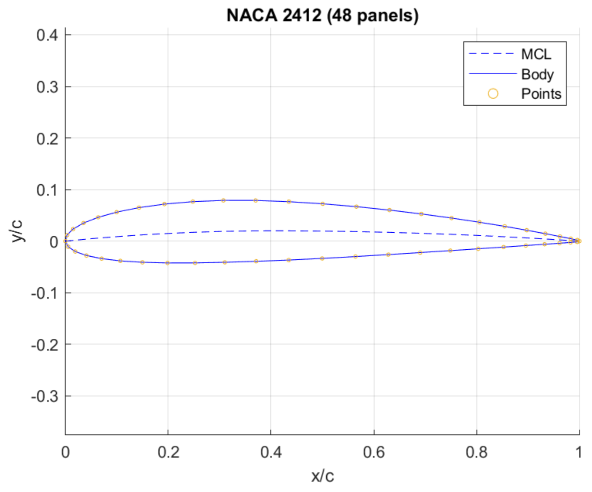
## IX. Appendix B – MATLAB Output

### Body Panel Geometry

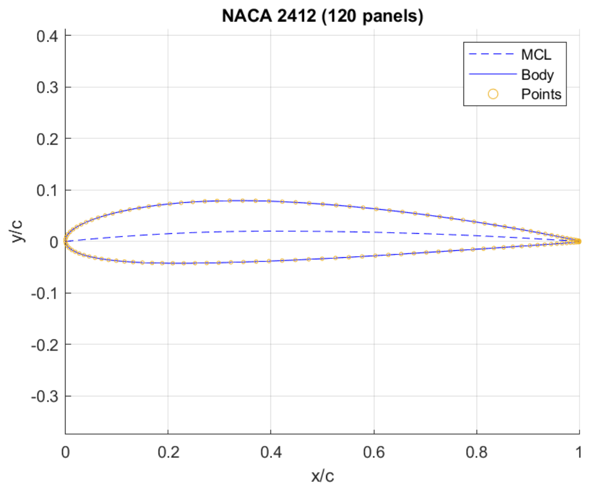
*1.1 Body 12 Panels*



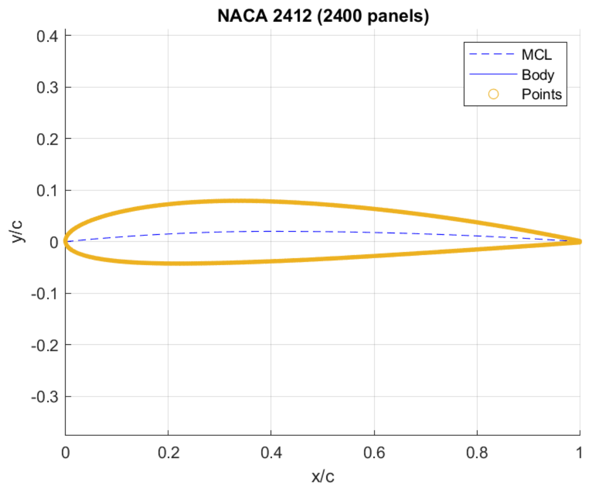
*1.2 Body 48 Panels*



*1.3 Body 120 Panels*

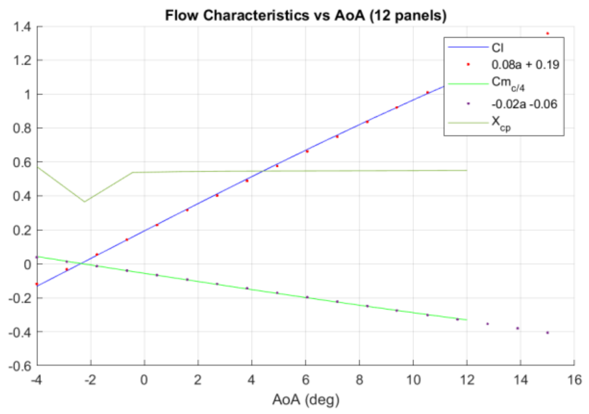


*1.4 Body 2400 Panels*

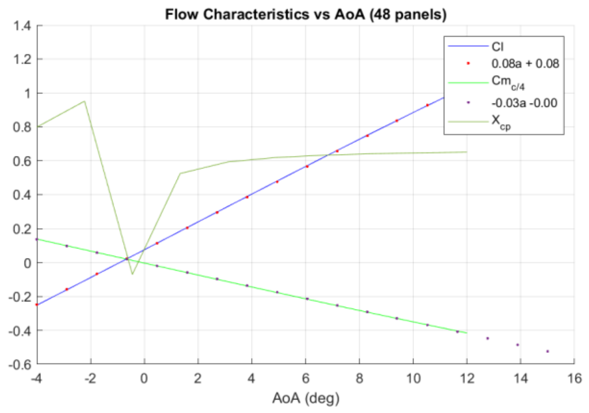


### 1. Flow Characteristics vs AoA

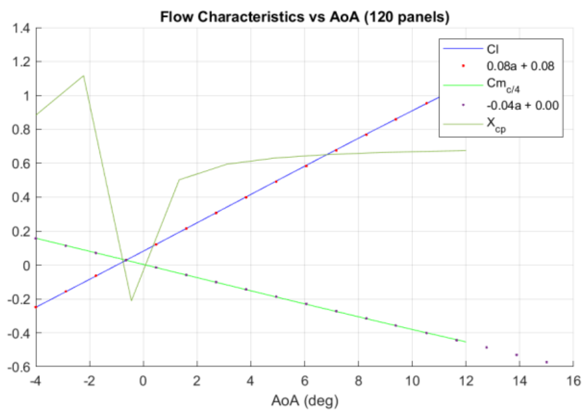
*2.1 Flow Characteristics 12 Panel*



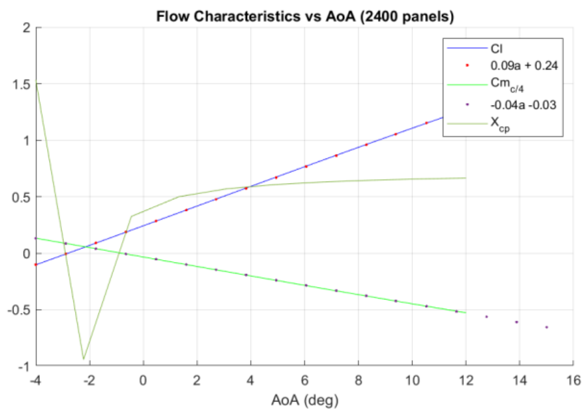
*2.2 Flow Characteristics 48 Panel*



*2.3 Flow Characteristics 120 Panel*

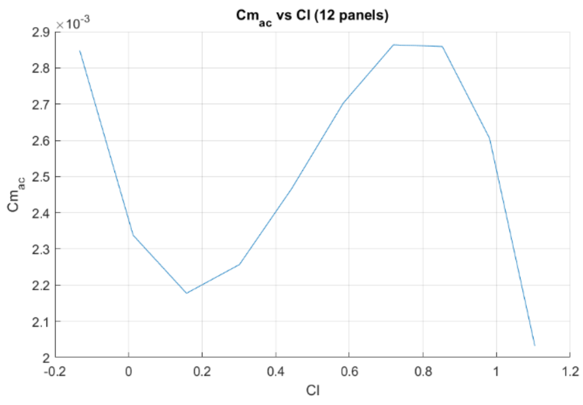


*2.4 Flow Characteristics 2400 Panel*

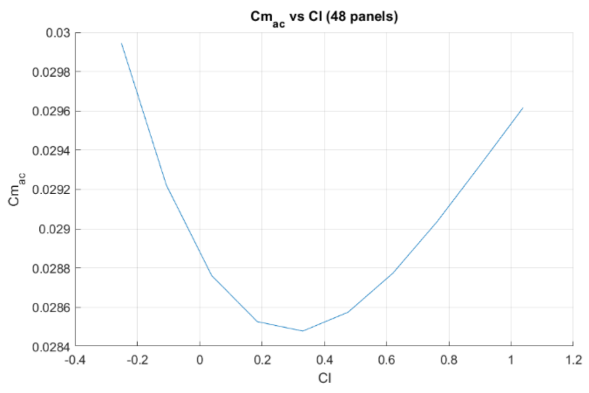


## 2. Cmac vs Cl

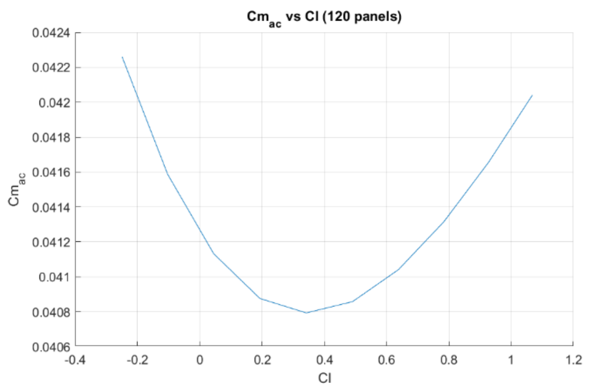
*3.1 Cmac vs Cl 12 Panels*



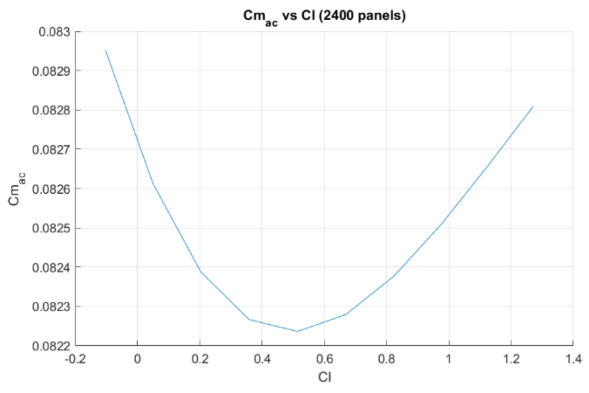
*3.2 Cmac vs Cl 48 Panels*



*3.3 Cmac vs Cl 120 Panels*



*3.4 Cmac vs Cl 2400 Panels*



## X. Appendix C – MATLAB Code

```matlab
%% Patrick Kennedy - Term Project
clear, clc, close all; clf;
set(0, 'DefaultFigureWindowStyle', 'docked');
should_plot = false;
airfoil = '2412';
V = 1;
m = 120;

%% Model NACA 2412 body
figure(1);
title(sprintf('NACA 2412 (%d panels)',m));
xlabel('x/c');
ylabel('y/c');
hold on; grid on; axis equal;
xlim([0 1]);

% read airfoil
naca2412 = naca4digit(airfoil, m);

% plot mcl and body
plot(naca2412.x, naca2412.mcl, 'b--'); % lower
plot(naca2412.XY(1,:), naca2412.XY(2,:), 'b-'); % body
scatter(naca2412.XY(1,:), naca2412.XY(2,:), 5); % vertecies
legend('MCL', 'Body', 'Points');

%% iterate accross AoAs
alphas = linspace(-4, 12, 10); %deg
output = cell(size(alphas));
newfig = 1;
for i = 1:length(alphas)
    a = alphas(i); % deg
    newfig = newfig + 1;
    if ~should_plot
        newfig = -1;
    end
    output{i} = vortext_panel_method(naca2412, a, newfig);
    fprintf([ ...
        'AoA %.2f\n\t' ...
        'Cl = %f\n\t' ...
        'Cd = %f\n\t' ...
        'Cmle = %f\n\t' ...
        'xcp = %f\n\t' ...
        'cmc4 = %f\n\n'], a, output{i}.cl, output{i}.cd, output{i}.cmle, ...
                        output{i}.xcp, output{i}.cmc4);
end

%% plot Cl vs a

plot_legend = ["",""];

Cl = zeros(size(output));
Cmc4 = zeros(size(output));
for i = 1:length(output)
    Cl(i) = output{i}.cl;
    Cmc4(i) = output{i}.cmc4;
end
```

```

newfig = newfig + 1;
if ~should_plot
    newfig = 2;
end

figure(newfig);
grid on; hold on;
title(sprintf('Flow Characteristics vs AoA (%d panels)', m));
xlabel('AoA (deg)');
plot(alphas, Cl, 'b-');
coeffs = polyfit(alphas, Cl, 1);
fplot(@(a) coeffs(1)*a + coeffs(2), [-4 15], 'r.');

if coeffs(2) >= 0
    plot_legend(end+1) = 'Cl';
    plot_legend(end+1) = sprintf('%.2fa + %.2f',coeffs);
else
    plot_legend(end+1) = 'Cl';
    plot_legend(end+1) = sprintf('%.2fa %.2f',coeffs);
end

% get slope
ao = coeffs(1);
a_Cl0 = -coeffs(2) / coeffs(1);
Cl_a0 = coeffs(2);
fprintf('a0 = %f\n', ao);
fprintf('a_Cl0 = %f\n', a_Cl0);
fprintf('Cl_a0 = %f\n', Cl_a0);

%% plot Cmc/4 vs a
plot(alphas, Cmc4, 'g-');
coeffs = polyfit(alphas, Cmc4, 1);
fplot(@(a) coeffs(1)*a + coeffs(2), [-4 15], '.');

if coeffs(2) >= 0
    plot_legend(end+1:end+2) = ["Cm_c_/_4", sprintf('%.2fa + %.2f',coeffs)];
else
    plot_legend(end+1:end+2) = ["Cm_c_/_4", sprintf('%.2fa %.2f',coeffs)];
end

% get slope
dcmc4_da = coeffs(1);

%% get aerodynamic center
xac = 0.25 - dcmc4_da/ao;
fprintf('x_ac = %f\n', xac);

%% plot xcp vs AoA
Xcp = zeros(size(output));
for i = 1: length(output)
   Xcp(i) = output{i}.xcp;
end

plot(alphas, Xcp);
legend([plot_legend(3:end) 'X_c_p']);
```

```

%% plot Cmac
Cmac = -Cl.*(Xcp - xac);
newfig = newfig + 1;
figure(newfig);
grid on; hold on;
plot(Cl, Cmac);
title(sprintf('Cm_a_c vs Cl (%d panels)',m));
xlabel('Cl');
ylabel('Cm_a_c');
coeffs = polyfit(Cl, Cmac, 0);
cmac = coeffs;
fprintf('Cmac fit = %f\n', cmac);

%% VORTEX PANEL METHOD ====================================================
function outs = vortext_panel_method(naca4, aoa_deg, fig)
%% parse parameters

alpha = deg2rad(aoa_deg);

if ~exist('fig','var')
    fig = -1;
end

num_panels = naca4.n;
Xbody = naca4.XY(1,:);
Ybody = naca4.XY(2,:);
num_points = num_panels + 1;

Ybody(1,1) = 0;
Ybody(end,1) = 0;

%% define geometry

X_mid = zeros(num_panels, 1); % x mid point
Y_mid = zeros(num_panels, 1); % y mid point
Sj = zeros(num_panels, 1); % length
theta = zeros(num_panels, 1); % panel angle

for i = 1:num_panels
    X_mid(i) = 0.5*( Xbody(i) + Xbody(i+1) );
    Y_mid(i) = 0.5*( Ybody(i) + Ybody(i+1) );
    Sj(i) = sqrt( (Xbody(i+1)-Xbody(i))^2 + (Ybody(i+1)-Ybody(i))^2 );
    theta(i,1) = atan2( (Ybody(i+1)-Ybody(i)) , (Xbody(i+1)-Xbody(i)) );
end

%% find coeffs

% init
Cn1 = zeros(num_panels);
Cn2 = zeros(num_panels);
Ct1 = zeros(num_panels);
Ct2 = zeros(num_panels);

for i = 1:num_panels
    for j = 1:num_panels
```

```

        if (i == j) % identity line

            Cn1(i,j) = -1;
            Cn2(i,j) = 1 ;
            Ct1(i,j) = pi/2;
            Ct2(i,j) = pi/2;

        else

            % current state
            sj = Sj(j);
            xi = X_mid(i);
            Xj = Xbody(j);
            thetai = theta(i);
            thetaj = theta(j);
            yi = Y_mid(i);
            Yj = Ybody(j);

            A = -(xi-Xj)*(cos(thetaj))-(yi-Yj)*(sin(thetaj));
            B = (xi-Xj)^2+(yi-Yj)^2;
            C = sin(thetai-thetaj);
            D = cos(thetai-thetaj);
            E = (xi-Xj)*sin(thetaj)-(yi-Yj)*cos(thetaj);
            F = log(1+(sj^2+2*A*sj)/B);
            G = atan2((E*sj),(B+A*sj));
            P = ((xi-Xj)*sin(thetai-2*thetaj))+((yi-Yj)*cos(thetai-2*thetaj));
            Q = ((xi-Xj)*cos(thetai-2*thetaj))-((yi-Yj)*sin(thetai-2*thetaj));

            Cn1(i,j) = 0.5*D*F+C*G-Cn2(i,j);
            Cn2(i,j) = D+((0.5*Q*F)/Sj(j))-((A*C+D*E)*(G/Sj(j)));
            Ct1(i,j) = 0.5*C*F-D*G-Ct2(i,j);
            Ct2(i,j) = C+((0.5*P*F)/Sj(j))+((A*D-C*E)*(G/Sj(j)));

        end
    end
end
clear sj xi Xj thetai thetaj yi Yj;

%% determine An, RHS, and At

% create An and RHS
An = zeros(num_points);
RHS = zeros(num_panels, 1);
for i = 1:num_points-1
    An(i,1) = Cn1(i,1);
    An(i,num_points) = Cn2(1,num_points-1);
    RHS(i) = sin(theta(i) - alpha);
    for j = 2:num_points-1
        An(i,j) = Cn1(i,j) + Cn2(i,j-1);
    end
end
An(num_points,1) = 1;
An(num_points,num_points) = 1;
for j = 2:num_points-1
    An(num_points,j) = 0;
```

```
end
RHS(num_points) = 0;

% create At
At = zeros(num_panels, num_points);
for i = 1:num_points-1
    At(i,1) = Ct1(i,1);
    At(i,num_points) = Ct2(i,num_points-1);
    for j = 2:num_points-1
        At(i,j) = Ct1(i,j) + Ct2(i,j-1);
    end
end

%% determine Cp
Gamma = An\RHS;

% calc velocity across each panel
V = zeros(1,num_panels);
for i = 1:num_panels
    % do summation
    Vsum = 0;
    for j = 1:num_points
        Vsum = Vsum + At(i,j)*Gamma(j);
    end

    % set velocity
    V(i) = cos(theta(i)-alpha) + Vsum;
end

% calc coeff of pressure across each panel
Cp = zeros(1,num_panels);
for i = 1:num_panels
    Cp(i) = 1 - (V(i))^2;
end

%% numerical integration to find outputs

Cp_l = flip(Cp(1:num_panels/2));
X_l = flip(X_mid(1:num_panels/2));
Cp_u = Cp(num_panels/2+1:end);
X_u = X_mid(num_panels/2+1:end);

% normal coeff
cn = 0;
for i = 1:length(Cp_l)
    dx = naca4.x(i) - naca4.x(i+1);
    cn = cn + (Cp_l(i) - Cp_u(i))*dx;
end

% axial coeff
ca = 0;
for i = 1:length(Cp_l)
    dx = naca4.x(i) - naca4.x(i+1);
    dy_l = Ybody(i) - Ybody(i+1);
    dy_u = Ybody(num_panels/2+i+1) - Ybody(num_panels/2+i);
    ca = ca + (Cp_u(i)*dy_u/dx - Cp_u(i)*dy_l/dx)*dx;
```

```
end

% calculate cl and cd
ClCd = [cos(alpha) -sin(alpha) ; sin(alpha) cos(alpha)] * [cn ; ca];
cl = ClCd(1);
cd = ClCd(2);

% coeff moment about LE
cmle = 0;
for i = 1:length(Cp_l)-1

    xil = Xbody(i);
    xi1l = Xbody(i+1);
    dxl = xil - xi1l;
    xiu = Xbody(num_points - i+1);
    xi1u = Xbody(num_points - i);
    dxu = xiu - xi1u;

    cmle = cmle + ...
        ((Cp_u(i) + Cp_u(i+1))/2)*((xiu+xi1u)/2)*dxu - ...
        ((Cp_l(i) + Cp_l(i+1))/2)*((xil+xi1l)/2)*dxl;
end

% xcp
xcp = -cmle / cl;

% cmc/4
cmc4 = -cl*(xcp-0.25);

% generate outs struct
outs = struct();
outs.cl = cl;
outs.cd = cd;
outs.cmle = cmle;
outs.xcp = xcp;
outs.cmc4 = cmc4;

%% plot
if fig ~= -1 % if should plot
    figure(fig);
    hold on;
    plot(naca4.XY(1,:), naca4.XY(2,:), 'b-'); % body
    plot(X_u,Cp_u, 'g');
    plot(X_l,Cp_l, 'r');
    plot(X_u,Cp_u+Cp_l);
    xline(xcp);
    set(gca,'Ydir','reverse');
    xlabel('x/c');
    ylabel('Coefficient of Pressure');
    grid on;
    title(sprintf('NACA %s @ AoA = %d', naca4.name, aoa_deg));
    legend('Body','Upper','Lower','Cp_u+Cp_l');
end

end
```

```
%% NACA 4-DIGIT ============================================================

% naca4(mpxx) returns a characterized airfoil using A&V mean cord line
% equations. This function will parse 4 digit numbers and return a struct
% of the airfoil characteristics
%
% digits = string: format 'mpxx' that represents NACA 4 digits
% n = int: number of panels for XY model, will only produce even number of
% panels, any odd number will be round down with floor()
function naca4 = naca4digit(digits, n)
naca4 = struct(); % init struct
naca4.name = digits;
naca4.n = n;
m = str2double(digits(1)) / 100; % parse m
p = str2double(digits(2)) / 10; % parse p
xx = str2double(digits(3:4)) / 100; % parse xx

% add demensions to airfoil
naca4.m = m;
naca4.p = p;
naca4.xx = xx;
naca4.Rle = 1.1019 * xx^2; % nose radius

% Model MCL
syms x_
yt = xx*(1.4845.*sqrt(x_) - 0.63.*x_ - 1.758.*x_.^2 ...
    + 1.4215.*x_.^3 - 0.5075.*x_.^4); % thickness distrubution
dyc_dx = diff(yt,x_);

% convert yt and dyc/dt to function handel
yt = matlabFunction(yt);
dyc_dx = matlabFunction(dyc_dx);

% add to airfoil
naca4.yt = yt;
naca4.dyt_dx = dyc_dx;

% mcl equations for fwd and aft
mcl_fwd = (2*m/p).*x_ - (m/p^2) .* x_.^2;
mcl_aft = (-m/(p^2 -2*p +1))*(2*p-1 - 2*p.*x_ + x_.^2);

% account for singularity
if p == 0
    mcl_fwd = 0;
end

% find derivate of y_mcl(x)
dmcl_fwd = diff(mcl_fwd, x_);
dmcl_aft = diff(mcl_aft, x_);

% convert syms to function handel
mcl_fwd = matlabFunction(mcl_fwd,'Vars',x_);
mcl_aft = matlabFunction(mcl_aft,'Vars', x_);
dmcl_fwd = matlabFunction(dmcl_fwd,'Vars',x_);
dmcl_aft = matlabFunction(dmcl_aft,'Vars',x_);

```

```
% conditional f(x) for mcl and dyc/dx
mcl = @(x) (x < p) .* mcl_fwd(x) ... % fwd
        + ~(x < p) .* mcl_aft(x); % aft

dyc_dx = @(x) (x < p) .* dmcl_fwd(x) ... % fwd
           + ~(x < p) .* dmcl_aft(x); % aft

% iterate from theta = 0 to 180 and collect x locations on r=0.5 circle
thetas = linspace(0, 180, floor(n/2)+1);
x = zeros(1,length(thetas));
r = 0.5;
for i = 1:length(thetas)
    theta = thetas(i);
    x(i) = r + r*cosd(theta);
end

% format x coords for upper and lower surface
xu = flip(x);
xl = x;

% trim x coords from LE and TE
xu = xu(2:end-1);
xl = xl(2:end-1);

% define body coords
Xu = @(x) x - yt(x).*sin(atan(dyc_dx(x)));
Xl = @(x) x + yt(x).*sin(atan(dyc_dx(x)));
Yu = @(x) mcl(x) + yt(x).*cos(atan(dyc_dx(x)));
Yl = @(x) mcl(x) - yt(x).*cos(atan(dyc_dx(x)));

% parse into vec of coords
% add (1, 0) in front and back, and (0, 0) in between upper and lower
XY = [1 Xl(xl) 0 Xu(xu) 1 ; % TE -> lower -> LE -> upper -> TE
      0 Yl(xl) 0 Yu(xu) 0]; % TE -> lower -> LE -> upper -> TE

% add values to airfoil
naca4.x = x;
naca4.mcl = mcl(x);
naca4.XY = XY;

end

```