Homework 3

To:	Andrew Ning
From:	Joshua Canlas
Date:	February 26, 2020
Subject:	Homework 3 Memo

Introduction

The objective of this homework was to find the boundary layer properties (δ^*, H, c_f , θ) of a flat plate and a NACA 2412 airfoil. In addition, we were also asked to find the coefficient of drag for the airfoil. For the flat plate, we compared the results we found using Thwaite's method (for laminar) and Head's method (for turbulent) to the Blasius and Schlichting solutions respectively. For the airfoil, we computed the coefficient of drag by adding the boundary layer properties to the solutions we got from the Hess/Smith panel method. The results that we found were verified with a NACA report (see Appendix C).

3.1 Flat Plate

Introduction

The objective of this problem was to find the displacement thickness (δ^*), momentum thickness (θ), shape factor (H), local coefficient of skin friction drag (c_f), and overall coefficient of skin friction drag (C_f) on a flat plate using Thwaite's method for laminar flow and Head's for turbulent (see Appendix A for code). We then compared the values found from Thwaite's method to the Blasius solutions and Head's with Schlichting's. We assumed all laminar flow when using Thwaite's method and all turbulent flow when using Head's.

Methods

To begin, we calculated the boundary layer properties using the Blasius and Schlichting solutions. The Reynolds number and initial conditions are defined as

$$Re_x = \frac{\rho V x}{\mu}$$

$$Re_x = 1 \times 10^6$$

$$x_0 = 1 \times 10^{-6}$$

$$H_0 = 1.4$$

$$\frac{dV_e}{dx} = 0$$

$$V_e = 10$$

where ρ is the density of air, μ is the dynamic viscosity, and V_e is the edge velocity.

Blasius:

The Blasius solutions were found using the following equations:

$$\delta^* = \frac{1.72x}{\sqrt{Re_x}}$$

$$\theta = \frac{0.664x}{\sqrt{Re_x}}$$

$$H = \frac{\delta^*}{\theta}$$

$$c_f = \frac{0.664}{\sqrt{Re_x}}$$

$$C_f = \frac{1}{L}\int_0^L c_f dx$$

Schlichting:
The Schlichting solutions were found with these equations:

$$\delta^* = \frac{0.046x}{Re_x^{0.2}}$$

$$\theta = \frac{0.036x}{Re_x^{0.2}}$$

$$H = \frac{\delta^*}{\theta}$$

$$c_f = \frac{0.0592x}{Re_x^{0.2}}$$

$$C_f = \frac{0.074x}{Re_x^{0.2}}$$

Thwaite's:
Thwaite's method requires solving an ODE in order to solve for all of the other boundary layer properties

$$\frac{d\theta}{dx} = \frac{0.225v}{V_e\theta} - \frac{3\theta}{V_e}\frac{dV_e}{dx}$$

The initial condition that we used to solve the ODE was the first solution from the Blasius solutions.
In order to solve for l and H, λ must first be computed:

$$\lambda = \frac{\theta^2}{v}\frac{dV_e}{dx}$$

$$\left.\begin{aligned} l &= 0.22 + 1.57\lambda - 1.8\lambda^2 \\ H &= 2.61 - 3.75\lambda - 5.24\lambda^2 \end{aligned}\right\} \text{ if } 0 \le \lambda \le 0.1$$

$$\left.\begin{aligned} l &= 0.22 + 1.402\lambda + \frac{0.018\lambda}{0.107 + \lambda} \\ H &= 2.088 + \frac{0.0731}{0.14 + \lambda} \end{aligned}\right\} \text{ if } -0.1 < \lambda < 0$$

Once the all of the H values were found, we used them to calculate δ^*

$$\delta^* = H\theta$$

The final values to determine were the local coefficients of skin friction and overall coefficient of skin friction

$$c_f = \frac{2vl}{\theta V_e}$$

$$C_f = \frac{1}{L}\int_0^L c_f dx$$

All of the desired values were found for each 'x' value with the exception of C_f , which is for the entire plate.

Head's:
This method required solving two ODEs simultaneously

$$\frac{dH_1}{dx} = \frac{0.0306}{\theta}(H_1 - 3)^{-0.6169} - \frac{dV_e}{dx}\frac{H_1}{V_e} - \frac{d\theta}{dx}\frac{H_1}{\theta}$$

$$\frac{d\theta}{dx} = \frac{c_f}{2} - \frac{dV_e}{dx}\frac{\theta}{V_e}(H + 2)$$

The initial value for θ was found using the first solution from Schlichting's solutions. The initial value for c_f was found using the following equations:

$$c_f = 0.246 \times 10^{-0.678H} Re_\theta^{-0.268}$$

$$Re_\theta = \frac{V_e\theta}{v}$$

The initial value for H_1 was determined using H = 1.28 as the initial condition in the following piecewise function:

$$H_1 = \begin{cases} 0.8234(H-1.1)^{-1.287} + 3.3 & H \le 1.6 \\ 1.5501(H-0.6778)^{-3.064} + 3.3 & H > 1.6 \end{cases}$$

Every iteration the H_1 value is then used to solve for H using this piecewise function:

$$H = \begin{cases} 0.86(H_1-3.3)^{-0.777} + 1.1 & H_1 \ge 5.3 \\ 1.1538(H_1-3.3)^{-0.326} + 0.6778 & H_1 < 5.3 \end{cases}$$

Once we solved for θ and H_1, we used the following equation to solve for the displacement thickness at each point on the airfoil:

$$\delta^* = H\theta$$

Finally, the overall coefficient of skin drag friction was calculated using this equation:

$$C_f = \frac{1}{L}\int_0^L c_f dx$$

Results and Discussion

We've plotted the results from each method and the solutions from Blasius and Schlichting on Figures 1 and 2. Since both Thwaite's method and the Blasius solutions predicted the values in laminar flow (which has better solutions than turbulent flow), their results were very similar. Similarly, Head's method and Schlichting's solutions both predicted values in a turbulent flow and were similar to each other.

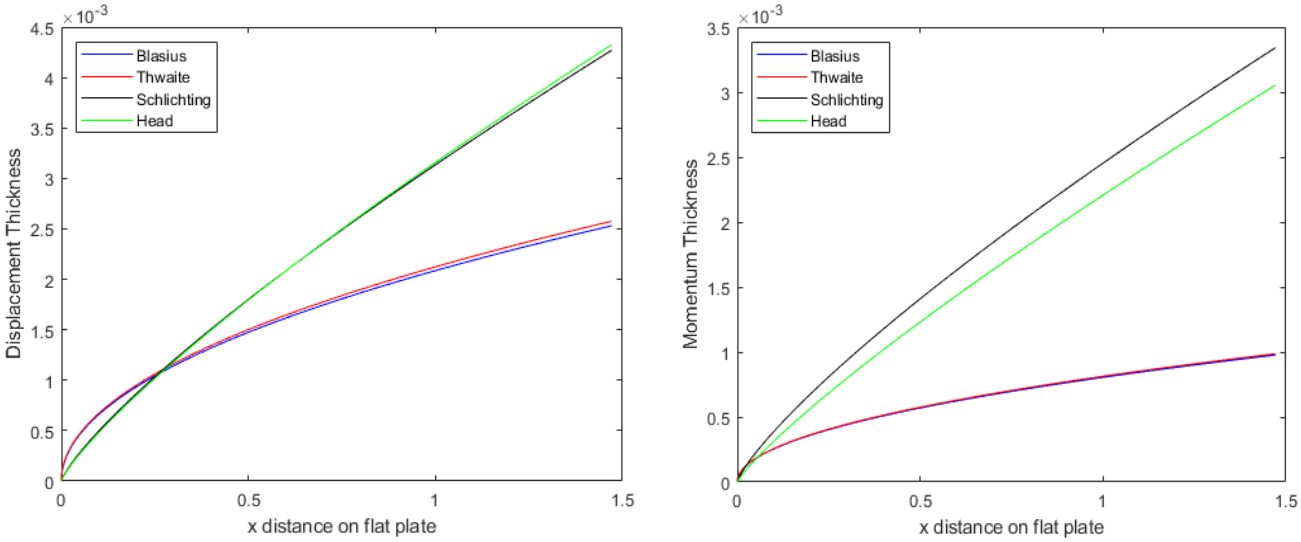


Figure 1: (a) Plot of displacement thickness vs. distance on a flat plate. (b) Plot of momentum thickness vs. distance on a flat plate. The results for all four methods are shown using different colors.

As can be seen in Fig. 1a, the displacement thickness results lined up nicely for both laminar and turbulent methods. As was mentioned in the course notes, the displacement thickness of the

laminar region grows faster initially, but then the turbulent region ends up growing at a much faster rate over the remainder of the plate.

The momentum thickness was compared in Fig. 1b. The laminar results steadily increased, while the turbulent results increased at a much faster rate. As can be seen, the turbulent results were not as close to each other as the laminar results were, which shows the difficulty involved with modeling turbulent flow accurately. Additionally, the results showed an increase in momentum thickness across the plate's length for both laminar and turbulent situations.

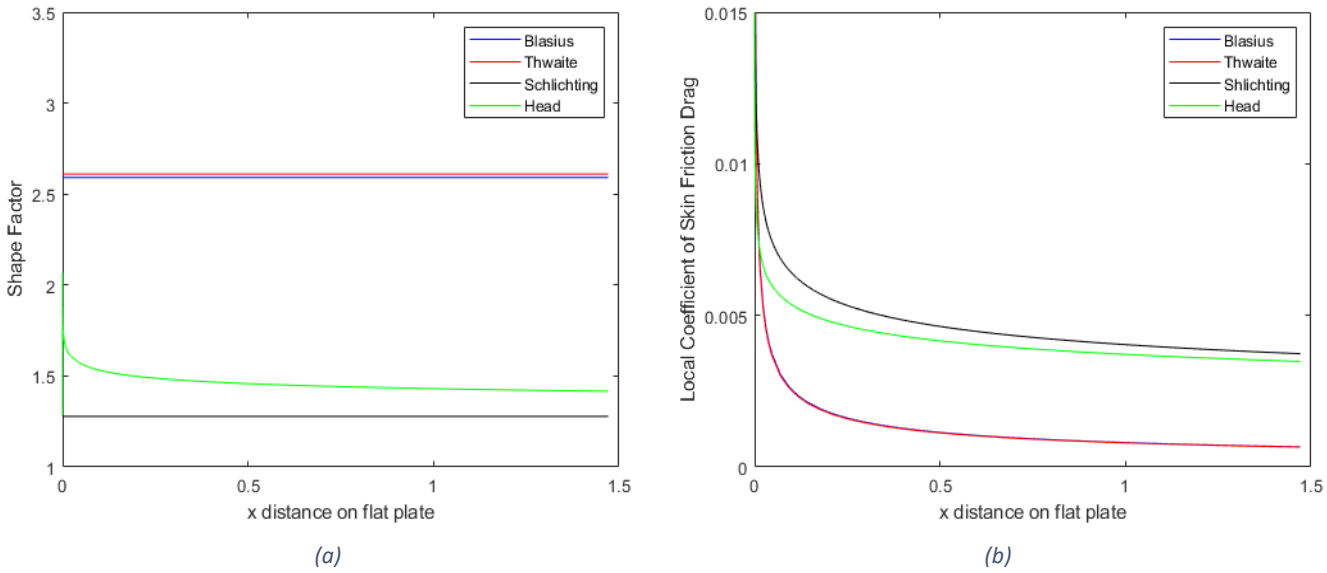


Figure 2: (a) Plot of shape factor vs. distance along a flat plate. (b) Plot of local coefficient of skin friction drag against length of the flat plate.

Fig. 2a shows the shape factor against the distance across the length of the plate. While the laminar solutions were very steady, Head's method had a spike near the leading edge of the plate. This was explained by the unsteady nature of turbulent flow and the difficulties in modeling it at the leading edge. However, as the shape factor of the head's method was observed along the distance of the plate, it settled to a value of about 1.5.

The final plot (Fig. 2b) showed the local coefficient of skin friction drag against the distance along the flat plate. This coefficient starts by going to infinity at the leading edge and then settling at around 0.005 for turbulent flow and 0.001 for laminar flow. Once again, the laminar solutions were nearly identical while the turbulent solutions were slightly different.

Table 1: Table of overall coefficients of skin drag for each

Method	Coefficient of Skin Drag Friction
Blasius	0.0014
Thwaite	0.0013
Schlicthting	0.0047
Head	0.0042

The overall coefficients of skin friction determined by each method are shown in Table 1. We verified the solutions that we calculated using Thwaite's and Head's by comparing them to the Blasius and Schlichting solutions. The differences between the solutions only vary by a small amount.

3.2 Airfoil

Introduction

The objective of this problem was to calculate the boundary layer properties of an airfoil and its coefficient of drag with an angle of attack at 5 degrees and with a Reynolds number of 1 x 10^6. We used the tangential velocities that we solved using the Hess/Smith panel method from the previous homework and used Thwaite's and Head's methods to find the boundary layer properties (see Appendix B). The values that we calculated using these methods were compared to results found in a NACA report (see Fig. C.1 in Appendix C).

Methods

The tangential velocities that we found using the Hess/Smith panel method from last homework were used in Thwaite's and Head's methods to solve for the boundary layer properties. The implementation of the Hess/Smith panel method can be found in Appendix D. We split up the airfoil into an upper and lower surface (with the stagnation point as the starting point) and passed the edge velocity values and lengths between each control point into Thwaite's. We solved for the boundary layer properties only until the boundary starts to transition. Once it began to transition, we switched over to Head's method. The criteria that we used for the transition period were

$$2.1 < H < 2.8$$

$$\log_{10}(Re_x) > -40.4557 + 64.8066H - 26.7538H^2 + 3.3819H^3$$

Using these criteria, we calculated the points where the flow transitioned from laminar to turbulent. At these points, we used the final values of the boundary layer properties from Thwaite's method as the initial values for Head's method (turbulent flow).

We assumed that the flow began to separate from the airfoil when the value of the shape factor was about 3. Once we found all of the values for the boundary layer properties, we calculated the coefficient of drag of the upper and lower surfaces of the airfoil and added them together to get the total coefficient of drag.

$$c_d = \frac{2\theta_{TE}}{c}\left(\frac{V_{eTE}}{V_\infty}\right)^{\frac{H_{TE}+5}{2}}$$

Results and Discussion

The displacement thickness, momentum thickness, shape factor, and local coefficient of skin friction for the upper and lower airfoil surfaces were plotted against distance from the stagnation point. Additionally, the transition points for each surface are located and plotted on each graph.

Fig. 3 shows the displacement thickness values for the upper and lower surfaces across airfoil. As shown, the upper surface transitions from laminar flow to turbulent flow much faster than the lower surface. After the flow transitions from the laminar to turbulent flow, the displacement thickness starts growing at an increasing rate that goes to infinity near separation. It makes sense that the transition point for the upper surface occurs earlier than the lower surface because the edge velocity is bigger in the upper surface.

Next, the momentum thickness was plotted against distance from the stagnation point (Fig. 4). This plot shows that the momentum thickness is continuous between laminar and turbulent flow, which is something that was predicted from the class notes. While the momentum thickness does not drop suddenly at the transition point, it does begin to grow more quickly in turbulent flow. This makes sense because there is more momentum in turbulent flow than in laminar. This value also approaches infinity as it nears separation which theoretically makes sense due to the loss of attachment of the boundary layer on the surface.

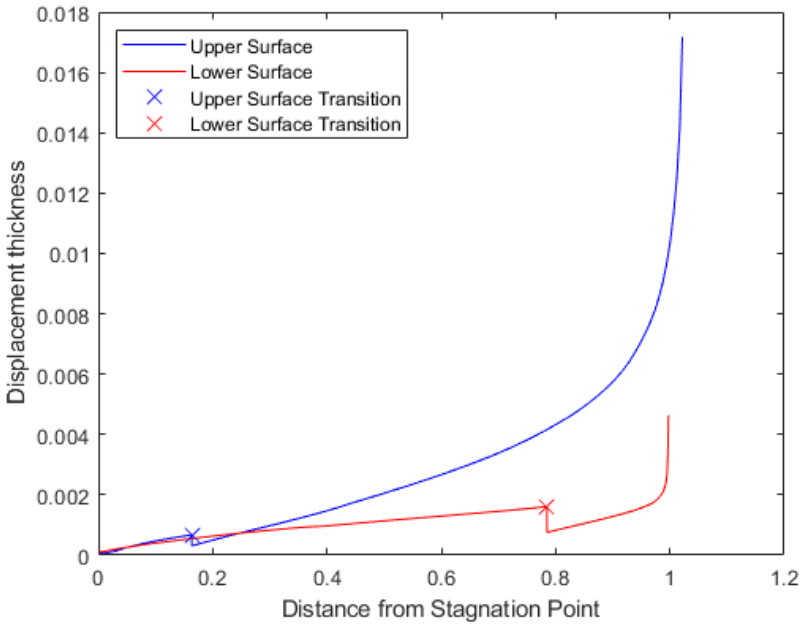


Figure 3: Displacement thickness against distance from the leading-edge stagnation point for upper and lower NACA 2414 airfoil surfaces.

The shape factor for the upper and lower surfaces of the NACA 2412 airfoil was plotted against distance from the stagnation point in Fig. 5. The shape factor is relatively high in the laminar regions for both surfaces, but then drops significantly when it reaches turbulence, as the notes in the course materials indicate. This is true because the shape factor is calculated by dividing the displacement thickness by the momentum thickness. When the flow transitions from laminar to turbulent, the displacement thickness stays about the same value, whereas the momentum thickness increases. Once turbulence has been reached, the shape factor then begins to increase again, going to infinity at separation.

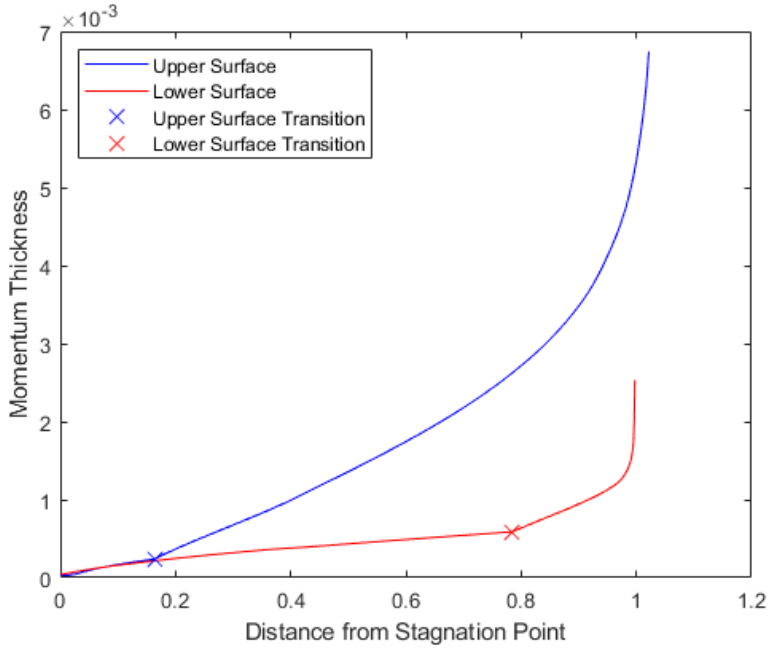


Figure 4: Momentum thickness against distance from the leading-edge stagnation point for upper and lower surfaces on NACA 2412 airfoil.

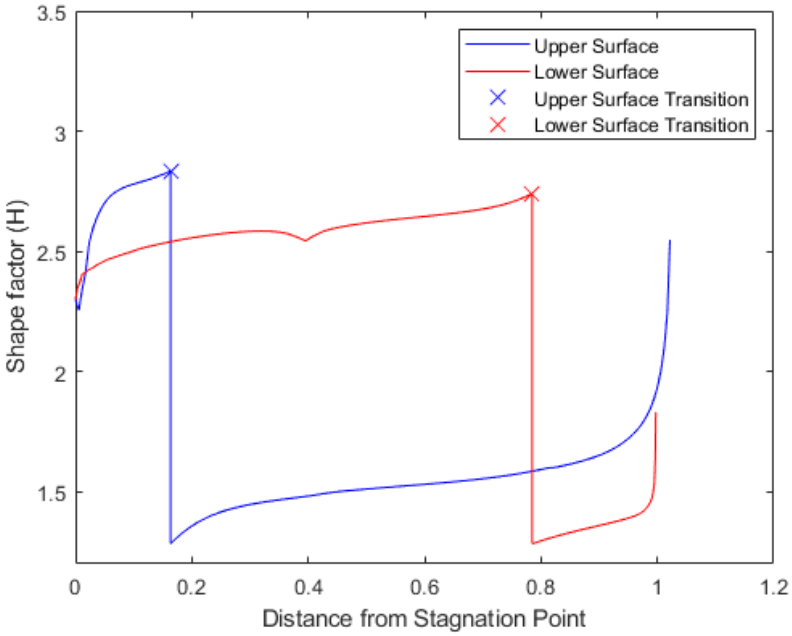


Figure 5: Shape factor against distance from the leading-edge stagnation point for the upper and lower surfaces of a NACA 2412 airfoil.

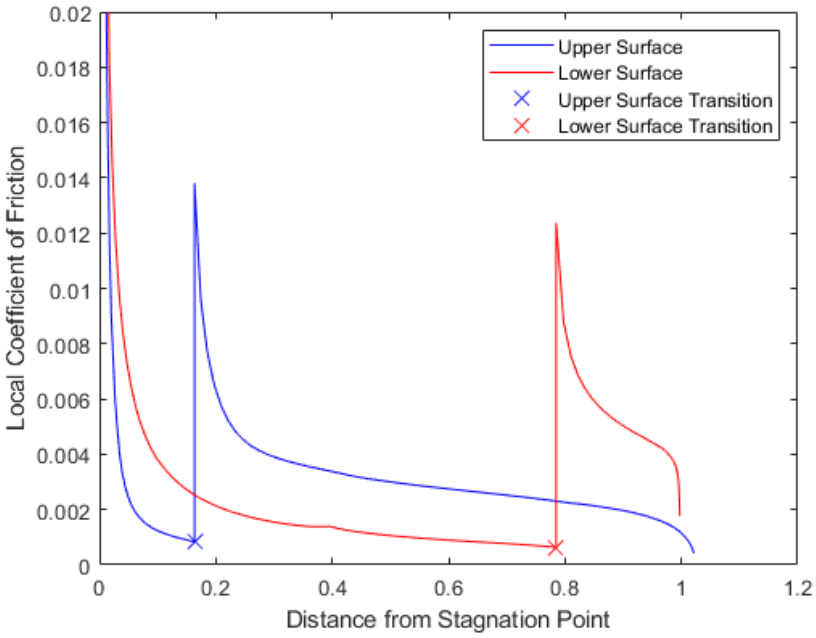


Figure 6: Local coefficient of skin drag friction against distance from the leading-edge stagnation point for the upper and lower surfaces of a NACA 2412 airfoil.

Finally, Fig. 6 shows the local coefficient of skin friction against the distance from the stagnation point. The coefficient of friction decreases along the distance until it reaches the transition point, at which it spikes up and then begins decreasing again. The coefficient then goes to zero as the flow separates. This makes sense intuitively because the coefficient of friction is highest at the stagnation point, and then should increase again when the airfoil trips turbulence.

We verified the results we found using these methods to the results found in the NACA report. The coefficient of drag that we calculated was about 0.005 and the one in the NACA report was about 0.009 (with the same Reynolds number and angle of attack). We expected the results to not be exactly the same, but we feel that our results are close enough to this experimental data that it is sufficient for the types of applications that we will encounter in this class.

Appendix A

Flat Plate:

```
clc
clear all

% Parameters
V_e = 10; % m/s
mu = 0.00001802;  % dynamic viscosity
rho = 1.225;
nu = mu / rho;  % kinematic viscosity
R_e = 1 * 10^6;
L = R_e*nu/V_e;  % length of plate
% N = 20;  % number of iterations
% x = L;
dVedx = 0;  % for flat plate

% Thwaite's method
x_0 = 10^-6;
R_e_x_0 = rho*V_e*x_0 / mu;
theta_0 = 0.664*x_0 / sqrt(R_e_x_0);

[t_thw, theta_thw] = ode45(@(x, theta_thw) dthetadx(x, theta_thw, nu, V_e,
dVedx), [x_0 L], theta_0);
gamma = theta_thw.^2 * dVedx ./ nu;

for R = 1:length(t_thw)
    if gamma(R) > 0.1
        gamma(R) = 0.1;
    end

    if gamma(R) >= 0 && gamma(R) <= 0.1
        ell_thw(R) = 0.22 + 1.57*gamma(R) - 1.8*gamma(R)^2;
        H_thw(R) = 2.61 - 3.75*gamma(R) - 5.24*gamma(R)^2;

    elseif gamma(R) > -0.1 && gamma(R) < 0
        ell_thw(R) = 0.22 + 1.402*gamma(R) + 0.018*gamma(R) ./ ...
            (0.107 + gamma(R));
        H_thw(R) = 0.2088 + 0.0731 / (0.14 + gamma(R));
    end

    delta_star_thw(R) = H_thw(R) * theta_thw(R);
    c_f_thw(R) = 2*nu*ell_thw(R) / (theta_thw(R)*V_e);
end

C_f_thw = 1/L * trapz(t_thw,c_f_thw);

% Blasius Solution for Flat Plate (laminar and incompressible)
R_e_bla = rho*V_e*t_thw ./ mu;

delta_star_bla = 1.72 .* t_thw ./ sqrt(R_e_bla);
theta_bla = 0.664 .* t_thw ./ sqrt(R_e_bla);
c_f_bla = 0.664 ./ sqrt(R_e_bla);
H_bla = delta_star_bla ./ theta_bla;
```

```
C_f_bla = 1/L * trapz(t_thw, c_f_bla);


% Head's Method
% initial conditions
R_e_x_0 = rho*V_e*x_0 ./ mu;
theta_head_0 = 0.036 * x_0 / R_e_x_0^0.2;
H1_0 = 10.6;

y0 = [theta_head_0, H1_0];

[t_head, y_head] = ode45(@(x, y_head) dydt(x, y_head, V_e, dVedx, nu), [x_0
L], y0);
theta_head = y_head(:, 1);
H1 = y_head(:, 2);
for R = 1:length(t_head)
    H_head(R) = H_solver(H1(R));
    c_f_head(R) = cf_solver(theta_head(R), H_head(R), V_e, nu);
    delta_star_head(R) = H_head(R) * theta_head(R);
end
C_f_head = 1/L * trapz(t_head,c_f_head);


% Schlichting Solutions
R_e_x_sch = rho*V_e.*t_head ./ mu;

delta_star_sch = 0.046 .* t_head ./ R_e_x_sch.^0.2;
theta_sch = 0.036 .* t_head ./ R_e_x_sch.^0.2;
c_f_sch = 0.0592 ./ R_e_x_sch.^0.2;
H_sch = delta_star_sch ./ theta_sch;

C_f_sch = 0.074 / (R_e_x_sch(length(t_head))^0.2);

%Plotting delta star
figure(1)
plot(t_thw,delta_star_bla,'b')
hold on
plot(t_thw,delta_star_thw,'r')
hold on
plot(t_head,delta_star_sch,'k')
hold on
plot(t_head,delta_star_head,'g')
legend('Blasius','Thwaite','Schlichting','Head', 'Location', 'northwest');
xlabel('x distance on flat plate');
ylabel('Displacement Thickness');
% title('Delta Star')


%-----------------------------------------------------------------
%Plotting theta
figure(2)
plot(t_thw,theta_bla,'b')
hold on
```

```
plot(t_thw,theta_thw,'r')
hold on
plot(t_head,theta_sch,'k')
hold on
plot(t_head,theta_head,'g')
legend('Blasius','Thwaite','Schlichting','Head', 'Location', 'northwest');
xlabel('x distance on flat plate');
ylabel('Momentum Thickness');
% title('Theta')


%-----------------------------------------------------------------
%Plotting H
figure(3)
plot(t_thw,H_bla,'b')
hold on
plot(t_thw,H_thw,'r')
hold on
plot(t_head,H_sch,'k')
hold on
plot(t_head,H_head,'g')
legend('Blasius','Thwaite','Schlichting','Head');
xlabel('x distance on flat plate');
ylabel('Shape Factor');
ylim([1 3.5])
% title('H')


%-----------------------------------------------------------------
%Plotting cf
figure(4)
plot(t_thw,c_f_bla,'b')
hold on
plot(t_thw,c_f_thw,'r')
hold on
plot(t_head,c_f_sch,'k')
hold on
plot(t_head,c_f_head,'g')
legend('Blasius','Thwaite','Shlichting','Head');
xlabel('x distance on flat plate');
ylabel('Local Coefficient of Skin Friction Drag');
% title('cf')
ylim([0 .015])


function y = dthetadx(x, theta, nu, V_e, dVedx) % for Thwaite's Method
    % y is new theta
    y = 0.225*nu / (V_e * theta) - 3*theta*dVedx / V_e;
end

function y = dydt(x, y_head, V_e, dVedx, nu) % for Head's Method
    % y_head(1) = theta, y_head(2) = H1
    y = zeros(2,1);
    H = H_solver(y_head(2));
    cf = cf_solver(y_head(1), H, V_e, nu);
```

```
    y(1) = cf/2 - dVedx*y_head(1)/V_e * (H + 2); % dthetadx
    temp = y_head(2) - 3;
    y(2) = (0.0306*(temp^-0.6169))/y_head(1) - ...
        dVedx*y_head(2)/V_e - (y(1)*y_head(2))/y_head(1); % dH1dx
end


function y = H_solver(H1)
    if H1 < 3.3
        y = 3.0;
    end
    temp = H1 - 3.3;
    if H1 >= 5.3
        y = 0.86*(temp^-0.777) + 1.1;
    elseif H1 < 5.3
        y = 1.1538*(temp^-0.326) + 0.6778;
    end
end

function y = cf_solver(theta, H, V_e, nu)
    R_e = V_e * theta / nu;
    y = 0.246 * 10^(-0.678*H) * R_e^-0.268;
end

function y = H1_solver(H)
    temp1 = H - 1.1;
    temp2 = H - 0.6778;
    if H <= 1.6
        y = 0.8234*(temp1^-1.287) + 3.3;
    elseif H > 1.6
        y = 1.5501*(temp2^-3.064) + 3.3;
    end
end
```

Appendix B

Airfoil:

```
clc
clear all

N = 200;

A = 2;
B = 4;
CC = 12;
chord = 1;

e = A / 100;
p = B / 10;
t = CC / 100;
phi = linspace(0, pi, N/2+1);
count = 1;
rho = 1.225; % kg/m^3
mu = 0.00001802;  % dynamic viscosity
nu = mu / rho;
R_e = 1 * 10^6;
V_inf = R_e*nu/chord;  % freestream velocity

alpha = 5 * (pi/180); % angle of attack in radians

x = 0.5*(1-cos(phi));

[T, ybar, dTdx, dybardx] = naca4(e, p, t, x);

y_up = ybar+T/2;
y_low = ybar-T/2;

x = [flip(x), x(2:end)];
y = [flip(y_low), y_up(2:end)];


% control points
for R = 1:size(x,2)-1
    x_bar(R) = (x(R) + x(R+1))/2;
    y_bar(R) = (y(R) + y(R+1))/2;
end

% Flow tangency boundary condition
for I = 1:N
    theta_i(I) = atan2((y(I+1) - y(I)) , ...
        (x(I+1) - x(I)));

    for J = 1:N
        r_1(I,J) = sqrt((x_bar(I)-x(J+1))^2 + ...
            (y_bar(I)-y(J+1))^2);

        theta_j(J) = atan2((y(J+1)-y(J)) , ...
```

```
            (x(J+1) - x(J)));

        r(I,J) = sqrt((x_bar(I)-x(J))^2 + ...
            (y_bar(I)-y(J))^2);


        if I ~= J
            const_1 = x(J)-x_bar(I);
            const_2 = y(J+1)-y_bar(I);
            const_3 = y(J)-y_bar(I);
            const_4 = x(J+1)-x_bar(I);

            beta(I,J) = atan2(((const_1*const_2) - ...
                            (const_3*const_4)) , ...
                            ((const_1*const_4) + ...
                            (const_3*const_2)));
        else
            beta(I,J) = pi;
        end
    end
end

for I = 1:N
    A(I, N+1) = 0.0;
    for J = 1:N
        A(I,J) = log(r_1(I,J)/r(I,J))*sin(theta_i(I)-theta_j(J)) + ...
            beta(I,J)*cos(theta_i(I)-theta_j(J));
        A(I,N+1) = A(I,N+1) + ...
            log(r_1(I,J)/r(I,J))*cos(theta_i(I)-theta_j(J)) - ...
            beta(I,J)*sin(theta_i(I)-theta_j(J));
    end
    b(I) = 2*pi*V_inf*sin(theta_i(I) - alpha);
end

k = 1;
A_k_sum = 0.0;
A_N_sum = 0.0;

% Kutta condition
for J = 1:N
    A_k = beta(k,J)*sin(theta_i(k)-theta_j(J)) - ...
        log(r_1(k,J)/r(k,J))*cos(theta_i(k)-theta_j(J));
    A_N = beta(N,J)*sin(theta_i(N)-theta_j(J)) - ...
        log(r_1(N,J)/r(N,J))*cos(theta_i(N)-theta_j(J));

    A_k_sum = A_k_sum + beta(k,J)*cos(theta_i(k)-theta_j(J)) + ...
        log(r_1(k,J)/r(k,J))*sin(theta_i(k)-theta_j(J));
    A_N_sum = A_N_sum + beta(N,J)*cos(theta_i(N)-theta_j(J)) + ...
        log(r_1(N,J)/r(N,J))*sin(theta_i(N)-theta_j(J));

    A(N+1,J) = A_k + A_N;
end
A(N+1, N+1) = A_k_sum + A_N_sum;
b(N+1) = -2*pi*V_inf * ...
    (cos(theta_i(k)-alpha)+cos(theta_i(N)-alpha));
```

```
q_gamma = inv(A)*b'; % solve for source and vortex strengths

% Solve for V_ti
for I = 1:N
    vel_src = 0.0;
    vel_vtx = 0.0;
    vel_free = V_inf*cos(theta_i(I)-alpha);

    for J = 1:N
        vel_src = vel_src + ...
            q_gamma(J)*(beta(I,J)*sin(theta_i(I)-theta_j(J)) - ...
            log(r_1(I,J)/r(I,J))*cos(theta_i(I)-theta_j(J)));
        vel_vtx = vel_vtx + ...
            q_gamma(N+1)*(beta(I,J)*cos(theta_i(I)-theta_j(J)) + ...
            log(r_1(I,J)/r(I,J))*sin(theta_i(I)-theta_j(J)));
    end

    V_ti(I) = vel_free + ...
        (1/(2*pi))*vel_src + ...
        (1/(2*pi))*vel_vtx;
end

% Calculate Cp
Cp = 1 - (V_ti/V_inf).^2;
% figure(2)
% plot(x_bar, -Cp, 'b')
% xlabel('x/c')
% ylabel('-Cp')

% Calculate Cm
P = 0.5*rho*V_inf^2.*Cp;
for R = 1:N
    F(R) = P(R)*sqrt((x(R+1)-x(R))^2+(y(R+1)-y(R))^2);
    x_dist(R) = 0.25 - x(R); % quarter chord
    y_dist(R) = 0.0 - y(R);
end
F_x = F.*sin(theta_i);
F_y = F.*cos(theta_i);
F = [F_x;F_y;zeros(1,length(F_x))]';
distance = [x_dist;y_dist;zeros(1,length(F_x))]';

moment = cross(F,distance,2);
moment = moment(:,3);
moment_sum = 0.0;
for R = 1:N
    moment_sum = moment_sum + moment(R);
end

Cm = moment_sum / (0.5*rho*V_inf^2*chord);

% Calculate Cl
gamma_sum = 0;
for R = 1:N
    dist = sqrt((x(R+1)-x(R))^2 + (y(R+1)-y(R))^2);
```

```
    gamma_sum = gamma_sum + (q_gamma(end)*dist);
end

lift = rho*V_inf*gamma_sum;
Cl = lift / (0.5*rho*V_inf^2*chord);

% Calculate Cd
% inviscid setup
[xu, Veu, xl, Vel] = parseairfoil(V_ti, x_bar, y_bar);
xu(1) = 10^-6;
xl(1) = 10^-6;
% initial conditions for Thwaite's
[Ve_upper_0, dVeudx_0] = velocity(xu, Veu, xu(1));
[Ve_lower_0, dVeldx_0] = velocity(xl, Vel, xl(1));

thetau_0_thw = sqrt((0.075*nu) / dVeudx_0);
thetal_0_thw = sqrt((0.075*nu) / dVeldx_0);


[t_u_thw, thetau_thw] =  ode45(@(x, thetau_thw) dthetadx(x, thetau_thw, nu,
xu, Veu), ...
                               xu, thetau_0_thw);
[t_l_thw, thetal_thw] =  ode45(@(x, thetal_thw) dthetadx(x, thetal_thw, nu,
xl, Vel), ...
                               xl, thetal_0_thw);

for R = 1:length(xu)
    [Ve_upper(R), dVeudx(R)] = velocity(xu, Veu, xu(R));
end

[ell_u_thw, H_u_thw, ds_u_thw, cf_u_thw] = calculate_thw(thetau_thw,
Ve_upper, ...
                                         dVeudx, t_u_thw, nu);

for R = 1:length(xl)
    [Ve_lower(R), dVeldx(R)] = velocity(xl, Vel, xl(R));
end

[ell_l_thw, H_l_thw, ds_l_thw, cf_l_thw] = calculate_thw(thetal_thw,
Ve_lower, ...
                                         dVeldx, t_l_thw, nu);

% Check for transition
for R = 1:length(H_u_thw)
    check_Re = -40.4557 + 64.8066*H_u_thw(R) - 26.7538*H_u_thw(R)^2 + ...
        3.3819*H_u_thw(R)^3;
    Re_x = Ve_upper(R) * xu(R) / nu;
    if H_u_thw(R) > 2.1 && log10(Re_x) > check_Re % && H_u_thw(R) < 2.8
        transu_idx = R;
        break;
    end
end

for R = 1:length(H_l_thw)
    check_Re = -40.4557 + 64.8066*H_l_thw(R) - 26.7538*H_l_thw(R)^2 + ...
```

```
        3.3819*H_l_thw(R)^3;
    Re_x = Ve_lower(R) * xl(R) / nu;
    if H_l_thw(R) > 2.1 && log10(Re_x) > check_Re && H_l_thw(R) < 2.8
        transl_idx = R;
        break;
    end
end

% initial conditions for Head's
thetau_0_head = thetau_thw(transu_idx);
thetal_0_head = thetal_thw(transl_idx);
H_one_0 = 10.6;

y0_upper = [thetau_0_head, H_one_0];
y0_lower = [thetal_0_head, H_one_0];


[t_u_head, y_u_head] = ode45(@(x, y_u_head) dydt(x, y_u_head,
xu(transu_idx:end), ...
                        Veu(transu_idx:end), nu), xu(transu_idx:end),
y0_upper);

[t_l_head, y_l_head] = ode45(@(x, y_l_head) dydt(x, y_l_head,
xl(transl_idx:end), ...
                        Vel(transl_idx:end), nu), xl(transl_idx:end),
y0_lower);

theta_u_head = y_u_head(:,1);
H1_u_head = y_u_head(:,2);
theta_l_head = y_l_head(:,1);
H1_l_head = y_l_head(:,2);

for R = 1:length(t_u_head)
    H_u_head(R) = H_solver(H1_u_head(R));
    cf_u_head(R) = cf_solver(theta_u_head(R), H_u_head(R), Veu(R), nu);
    ds_u_head(R) = H_u_head(R) * theta_u_head(R);
end

for R = 1:length(t_l_head)
    H_l_head(R) = H_solver(H1_l_head(R));
    cf_l_head(R) = cf_solver(theta_l_head(R), H_l_head(R), Vel(R), nu);
    ds_l_head(R) = H_l_head(R) * theta_l_head(R);
end

% separation
for R = 1:length(t_u_head)
    if H_u_head(R) >= 3
        sepu = R-1;
        break;
    else
        sepu = R;
    end
end
for R = 1:length(t_l_head)
    if H_l_head(R) >= 3
```

```
        sepl = R - 1;
        break;
    else
        sepl = R;
    end
end

% concatenate thwaite and head
% upper
H_u = [H_u_thw(1:transu_idx), H_u_head(1:sepu)];
theta_u = [thetau_thw(1:transu_idx)', theta_u_head(1:sepu)'];
delta_star_u = [ds_u_thw(1:transu_idx), ds_u_head(1:sepu)];
cf_u = [cf_u_thw(1:transu_idx), cf_u_head(1:sepu)];
t_u = [t_u_thw(1:transu_idx)', t_u_head(1:sepu)'];
% Cf_u = 1/chord * trapz(t_u, cf_u);

% lower
H_l = [H_l_thw(1:transl_idx), H_l_head(1:sepl-1)];
theta_l = [thetal_thw(1:transl_idx)', theta_l_head(1:sepl-1)'];
delta_star_l = [ds_l_thw(1:transl_idx), ds_l_head(1:sepl-1)];
cf_l = [cf_l_thw(1:transl_idx), cf_l_head(1:sepl-1)];
t_l = [t_l_thw(1:transl_idx)', t_l_head(1:sepl-1)'];
% Cf_l = 1/chord * trapz(t_l, cf_l);


Cd_u = (2*theta_u(end)/chord)*(Veu(end)/V_inf)^((H_u(end)+5)/2);
Cd_l = (2*theta_l(end)/chord)*(Vel(end)/V_inf)^((H_l(end)+5)/2);

Cd = Cd_u + Cd_l;

% plot boundary layer properties
figure(3)
plot(t_u, H_u, 'b', t_l, H_l, 'r', t_u(transu_idx), H_u(transu_idx), 'bx',
...
    t_l(transl_idx), H_l(transl_idx), 'rx', 'MarkerSize', 10)
xlabel('Distance from Leading Edge')
ylabel('Shape factor (H)')
ylim([1.2 3.5])
legend('Upper Surface', 'Lower Surface', 'Upper Surface Transition', 'Lower
Surface Transition')
figure(4)
plot(t_u, theta_u, 'b', t_l, theta_l, 'r', t_u(transu_idx),
theta_u(transu_idx), 'bx', ...
    t_l(transl_idx), theta_l(transl_idx), 'rx', 'MarkerSize', 10)
xlabel('Distance from Leading Edge')
ylabel('Momentum Thickness')
legend('Upper Surface', 'Lower Surface', 'Upper Surface Transition', 'Lower
Surface Transition', 'Location', 'northwest')
figure(5)
plot(t_u, delta_star_u, 'b', t_l, delta_star_l, 'r', t_u(transu_idx),
delta_star_u(transu_idx), 'bx', ...
    t_l(transl_idx), delta_star_l(transl_idx), 'rx', 'MarkerSize', 10)
xlabel('Distance from Leading Edge')
ylabel('Displacement thickness')
legend('Upper Surface', 'Lower Surface', 'Upper Surface Transition', 'Lower
Surface Transition', 'Location', 'northwest')
```

```
figure(6)
plot(t_u, cf_u, 'b', t_l, cf_l, 'r', t_u(transu_idx), cf_u(transu_idx), 'bx',
...
    t_l(transl_idx), cf_l(transl_idx), 'rx', 'MarkerSize', 10)
xlabel('Distance from Leading Edge')
ylabel('Local Coefficient of Friction')
ylim([0 0.02])
legend('Upper Surface', 'Lower Surface', 'Upper Surface Transition', 'Lower
Surface Transition')

%---------------FUNCTIONS---------------
function [T, ybar, dTdx, dybardx] = naca4(e, p, t, x)

    T = 10*t*(0.2969*sqrt(x) - 0.126*x - 0.3536*x.^2 + ...
        0.2843*x.^3 - 0.1015*x.^4);
    dTdx = 10*t*(0.2969*0.5./sqrt(x) - 0.126 - 0.3537*2*x + ...
        0.2843*3*x.^2 - 0.1015*4*x.^3);

    n = length(x);
    ybar = zeros(1, n);
    dybardx = zeros(1, n);

    for i = 1:n
        if x(i) <= p
            ybar(i) = e/p^2 * (2*p*x(i) - x(i)^2);
            dybardx(i) = e/p^2 * (2*p - 2*x(i));
        else
            ybar(i) = e/(1-p)^2 * (1 - 2*p + 2*p*x(i) - x(i)^2);
            dybardx(i) = e/(1-p)^2 * (2*p - 2*x(i));
        end
    end
end

%--------------Boundary Conditions-------------
function y = dthetadx(x, theta, nu, xv, Vev) % for Thwaite's Method
    % y is new theta
    [V_e, dVedx] = velocity(xv, Vev, x);
    y = 0.225*nu / (V_e * theta) - 3*theta*dVedx / V_e;
end

function y = dydt(x, y_head, xv, Vev, nu) % for Head's Method
    % y_head(1) = theta, y_head(2) = H1
    [V_e, dVedx] = velocity(xv, Vev, x);
    y = zeros(2,1);
    H = H_solver(y_head(2));
    cf = cf_solver(y_head(1), H, V_e, nu);

    y(1) = cf/2 - dVedx*y_head(1)/V_e * (H + 2); % dthetadx
    if y_head(2) < 3
        y(2) = 0;
    else
        temp = y_head(2) - 3;
        y(2) = (0.0306*temp^-0.6169)/y_head(1) - ...
            (dVedx*y_head(2))/V_e - (y(1)*y_head(2))/y_head(1); % dH1dx
    end
```

```
end


function y = H_solver(H1)
    temp = H1 - 3.3;

    if H1 < 3.3
        y = 3.0;
    elseif H1 >= 5.3
        y = 0.86*temp^-0.777 + 1.1;
    elseif H1 < 5.3
        y = 1.1538*temp^-0.326 + 0.6778;
    end
end

function y = cf_solver(theta, H, V_e, nu)
    R_e = V_e * theta / nu;
    y = 0.246 * 10^(-0.678*H) * R_e^-0.268;
end

function y = H1_solver(H)
    temp1 = H - 1.1;
    temp2 = H - 0.6778;
    if H <= 1.6
        y = 0.8234*(temp1)^-1.287 + 3.3;
    elseif H > 1.6
        y = 1.5501*(temp2)^-3.064 + 3.3;
    end
end

function [ell, H, delta_star, cf] = calculate_thw(theta, Ve, dVedx, t, nu)
    for R = 1:length(t)
        gamma = theta(R)^2 * dVedx(R) / nu;
        if gamma >= 0 && gamma <= 0.1
            ell(R) = 0.22 + 1.57*gamma - 1.8*gamma^2;
            H(R) = 2.61 - 3.75*gamma - 5.24*gamma^2;

        elseif gamma > -0.1 && gamma < 0
            ell(R) = 0.22 + 1.402*gamma + 0.018*gamma / ...
                (0.107 + gamma);
            H(R) = 2.088 + (0.0731 / (0.14 + gamma));
        elseif gamma > 0.1
            gamma = 0.1;
        else
            break; % laminar separation
        end
        delta_star(R) = H(R) * theta(R);
        cf(R) = 2*nu*ell(R) / (theta(R)*Ve(R));
    end
end

%-----------Ve and dVEdx---------------
function [xu, Veu, xl, Vel] = parseairfoil(Vt, x, y)
    % find stagnation point
    idx = find(Vt > 0, 1, 'first');
```

```
    % separate into upper and lower surfaces
    xu = x(idx:end);
    yu = y(idx:end);
    Veu = Vt(idx:end);

    xl = x(idx-1:-1:1);
    yl = y(idx-1:-1:1);
    Vel = -Vt(idx-1:-1:1);  %negative sign because of definition of Vt

    % compute distance along curved path around airfoil
    nu = length(xu);
    su = zeros(nu,1);
    for i = 1:nu-1
        su(i+1) = su(i) + sqrt((xu(i+1) - xu(i))^2 + (yu(i+1) - yu(i))^2);
    end

    nl = length(xl);
    sl = zeros(nl,1);
    for i = 1:nl-1
        sl(i+1) = sl(i) + sqrt((xl(i+1) - xl(i))^2 + (yl(i+1) - yl(i))^2);
    end

    % rename to x (boundary layer definition for x)
    xu = su';
    xl = sl';
end

function [Ve, dVedx] = velocity(xv, Vev, x)
    if x < xv(1)
        disp("provided x value is below xv range")
    elseif x > xv(end)
        disp("provided x value is above xv range")
    end

    % find index for linear interpolation
    idx = find(x >= xv, 1, 'last');
    if idx == length(xv)
        idx = idx - 1;
    end

    frac = (x - xv(idx)) / (xv(idx+1) - xv(idx));

    Ve = Vev(idx) +  frac*(Vev(idx+1) - Vev(idx));

    dVedx = (Vev(idx+1) - Vev(idx)) / (xv(idx+1) - xv(idx));
end
```

Appendix C

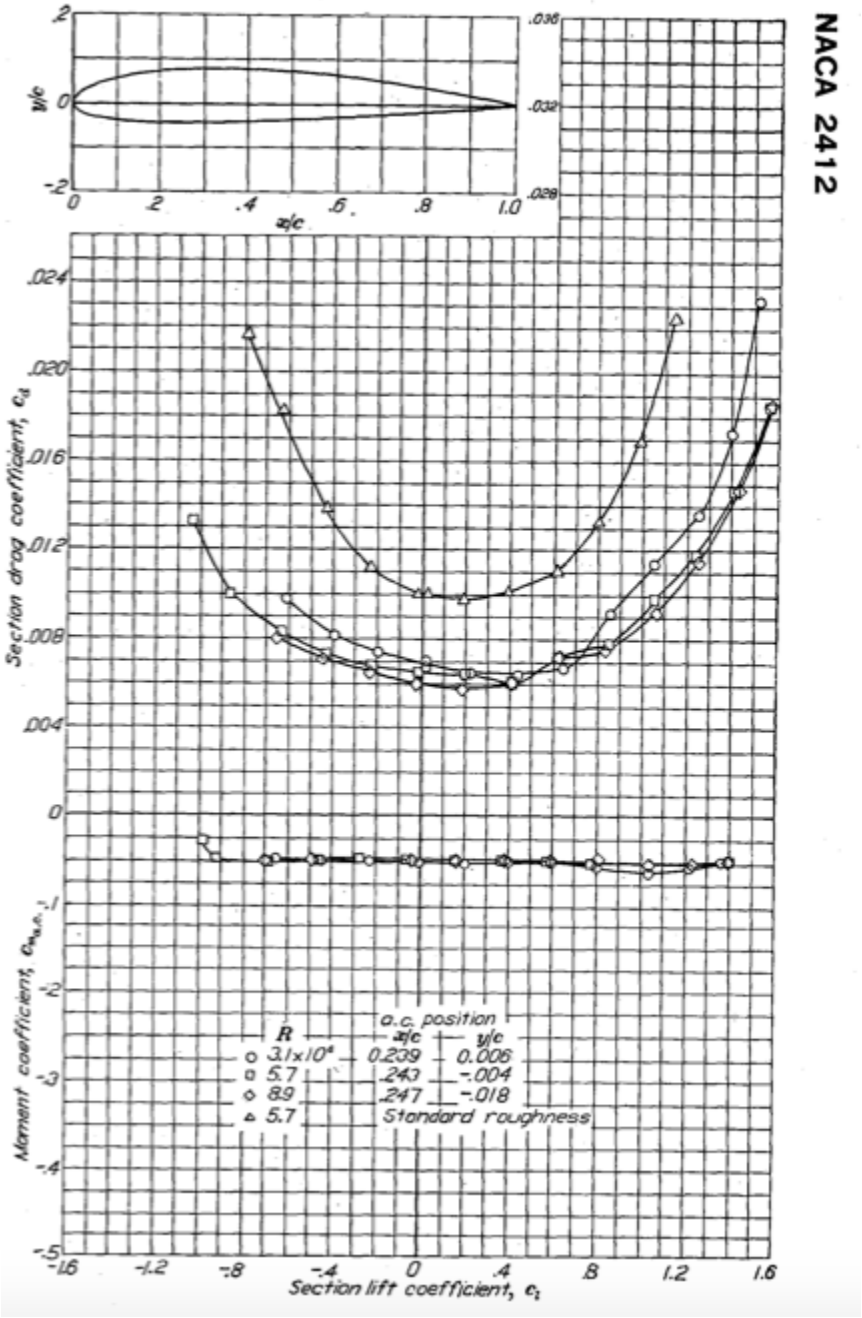


Figure C.1: This plot came from a summary of airfoil data from the National Advisory Committee for Aeronautics. The plot shows coefficient of lift versus coefficient of drag for various airfoils. The NACA 2412 airfoil corresponds to the line with circles on it.

Appendix D

To begin, a MATLAB code is created that splits the airfoil into many different panels by taking a finite number of points along the outside of the airfoil and connecting them with strait lines. The points are connected by going clockwise around the airfoil (keeping the body to the right of the panel) and then center points are found by taking an average of the x and y position of the two end points. These are then plotted on top of the airfoil which was provided in the code given by Dr. Ning.

To solve for the velocities, some simple geometry is first completed to determine the position of the various panels in relationship to each other. The following equations are used to relate the distance and angles between panel i (the panel being observed) and panel j (the panel influencing panel i).

$$r_{ij} = \sqrt{(\overline{x} - x_j)^2 + (\overline{y} - y_j)^2}$$

$$r_{ij+1} = \sqrt{(\overline{x} - x_{j+1})^2 + (\overline{y} - y_{j+1})^2}$$

$$\theta_i = \tan^{-1}\left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i}\right)$$

$$\theta_j = \tan^{-1}\left(\frac{y_{j+1} - y_j}{x_{j+1} - x_j}\right)$$

$$\beta_{ij} = \begin{cases} \text{atan2}\left(\frac{(x_j - \bar{x}_i)(y_{j+1} - \bar{y}_i) - (y_j - \bar{y}_i)(x_{j+1} - \bar{x}_i)}{(x_j - \bar{x}_i)(x_{j+1} - \bar{x}_i) + (y_j - \bar{y}_i)(y_{j+1} - \bar{y}_i)}\right) & \text{if } i \neq j \\ \pi & \text{if } i = j \end{cases}$$

Next, the source strengths and vortex strength are found using a matrix to combine multiple equations at once as shown below.

$$\begin{bmatrix} A_{11} & \cdots & A_{1N} & A_{1,N+1} \\ \vdots & & \vdots & \vdots \\ A_{N1} & \cdots & A_{NN} & A_{N,N+1} \\ A_{N+1,1} & \cdots & A_{N+1,N} & A_{N+1,N+1} \end{bmatrix} \begin{bmatrix} q_1 \\ \vdots \\ q_N \\ \gamma \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_N \\ b_{N+1} \end{bmatrix}$$

Where the following equations result from the flow tangency condition ($\vec{V} \cdot \hat{n} = 0$):

$$A_{ij} = \left[\ln\left(\frac{r_{ij+1}}{r_{ij}}\right)\sin(\theta_i - \theta_j) + \beta_{ij}\cos(\theta_i - \theta_j)\right]$$

$$A_{iN+1} = \sum_{j=1}^{N}\left[\ln\left(\frac{r_{ij+1}}{r_{ij}}\right)\cos(\theta_i - \theta_j) - \beta_{ij}\sin(\theta_i - \theta_j)\right]$$

$$b_i = 2\pi V_\infty \sin(\theta_i - \alpha)$$

Similarly, the following equations result from the Kutta condition:

$$A_{N+1,j} = \sum_{k=1 \text{ and } N} \left[\beta_{kj} \sin(\theta_k - \theta_j) - \ln\left(\frac{r_{kj+1}}{r_{kj}}\right) \cos(\theta_k - \theta_j) \right]$$

$$A_{N+1,N+1} = \sum_{k=1 \text{ and } N} \left(\sum_{j=1}^{N} \left[\beta_{kj} \cos(\theta_k - \theta_j) + \ln\left(\frac{r_{kj+1}}{r_{kj}}\right) \sin(\theta_k - \theta_j) \right] \right)$$

$$b_{N+1} = -2\pi V_\infty \left[\cos(\theta_1 - \alpha) + \cos(\theta_N - \alpha) \right]$$

Using these two conditions the entire matrix can be solved for q_n and γ. These values are then inserted into the tangential velocity equation.

$$V_{ti} = V_\infty \cos(\theta_i - \alpha) + \frac{1}{2\pi} \sum_{j=1}^{N} \left[q_j \left(\beta_{ij} \sin(\theta_i - \theta_j) - \ln\left(\frac{r_{ij+1}}{r_{ij}}\right) \cos(\theta_i - \theta_j) \right) \right] + \frac{\gamma}{2\pi} \sum_{j=1}^{N} \left[\beta_{ij} \cos(\theta_i - \theta_j) + \ln\left(\frac{r_{ij+1}}{r_{ij}}\right) \sin(\theta_i - \theta_j) \right]$$

This velocity can then be used to find the coefficient of pressure using the following equation:

$$C_p(\bar{x}_i, \bar{x}_j) = 1 - \left(\frac{V_{ti}}{V_\infty}\right)^2$$

Next, the coefficient of lift is found using this equation:

$$c_L = \frac{L'}{\frac{1}{2}\rho v^2 A}$$

$$L' = \rho V_\infty \times \Gamma$$

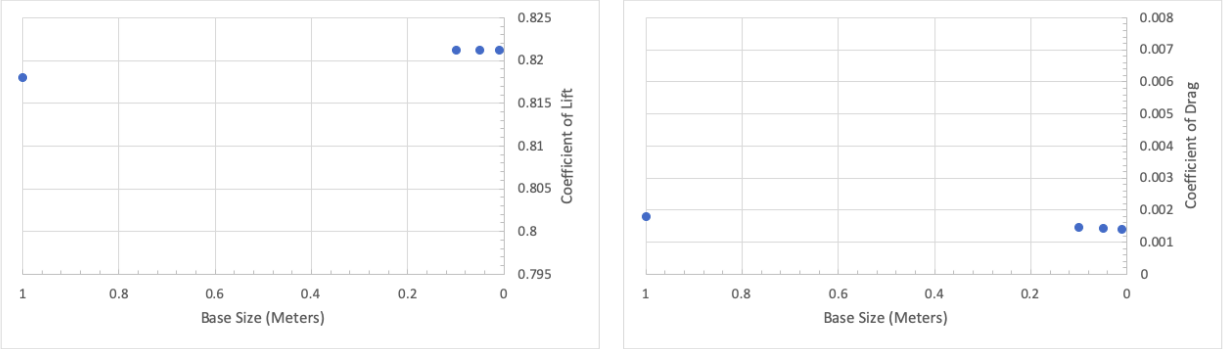
Where the total circulation is found to be the following equation since γ is known to be constant over all panels:

$$\Gamma = \int_0^c \gamma(s) ds$$

Therefore, the coefficient of lift is:

$$c_L = \frac{\rho V_\infty \Gamma}{\frac{1}{2}\rho v^2 c}$$

The coefficient of pitching moment is found using the following equation:



(a) *(b)*

Figure A3: (a) Coefficient of lift vs. base size in meters. The coefficient converges toward the value of 0.821 as base size decreases. (b) Coefficient of drag vs. base size in meters. The coefficient converges toward the value of 0.0014 as the base size.

$$c_m = \frac{M'}{\frac{1}{2}\rho v^2 Ac}$$

The moment is determined by finding the moment arms and forces on each panel caused by the pressure at the center points of each panel. Therefore, the equation for M' can be found using the following equation:

$$M' = \sum_0^n r_{ac} \times F_p$$