Ilia Kheirkhah

12/21/2022

**Table of Contents**



# MEAM 5450 Project 2 - Numeric Lifting Line

## Background

In class, we have discussed the numeric lifting line technique for wing analysis. Here, I will attempt to program this approach and see what results we get. Before we begin, let us recap the fundamental approach and theory.

The first important idea for numeric lifting line is the induced angle of attack. When we have a lifting wing, the vortex distribution causes there to be a change in effective angle of attack at an individual cross-section of the wing. Finding this induced angle of attack is the first fundamental hurdle of this method.

Once we have calculated the induced angle of attack, we can calculate our effective angle of attack. Whether by tables, linear aerodynamics, or some other theory, we can obtain the lift coefficient. From here, we can apply the kutta joukowski theorem and find the circulation. This relationship was provided in the textbook, and will be written here for ease of access

$$\Gamma_n = \frac{1}{2} V_{\infty,n} \cdot c_n \cdot c_{l,n}$$

One should not that this can vary with length and is unique at each cross-section.

Once we have this new circulation distribution, the final step is to adjust our circulation distribution. Although we could simply replace the old distribution with the new distribution, this would often lead to bad simulation results.

Instead, we introduce a damping factor $D$. This will allow us to slow down the simulation as to generate less non-converging results. This means that our change in circulation can be determined by:

$$\Delta\Gamma = D(\Gamma_{new} - \Gamma_{old})$$

Once we have this change, we can simply adjust our input circulation and rerun the simulation. with the following input

$$\Gamma_{\text{input}} = \Gamma_{\text{old}} + \Delta\Gamma$$

We repeat this process, adjusting our input circulation until it starts to converge on one value. We generally define convergence as $\Delta\Gamma$ being less than .01 for more than 5 consequtive iterations. This convergence criteria could be changed, but this is the main one mentioned in the textbook.

To summarize, we can break numeric lifting panels into  steps that repeat until convergence is reached:

1. From a given circulation distribution, calculate the induced and effective angles of attack.
2. From the effective angle of attack, calculate the new lift coefficient at each cross-section.
3. From the new lift coefficient distribution, calculate the lift per unit span and circulation distributions using the Kutta-Joukowski Theorem and Fundamental Lift Equation
4. With dampening, adjust the input circulation distribution and check for convergence. If convergence is not met, go back to step 1 with the adjusted input circulation. Repeat until convergence is met.

## Main Equations

For each section, there is a primary equation that governs the flow of the problem. Here we list those equations.

**1. Induced and effective angle of attack from Circulation**

$$V_{in} = \sum_{k=1}^{n} \frac{-\Gamma_t}{4\pi(y_p - y_t)}$$

Where the circulation we use here is for the

$$\alpha_{eff}(y_n) = \alpha - \alpha_i(y_n)$$

**2. Lift coefficient from effective Angle of Attack**

Dependent on section

$c_l = 2\pi\alpha$ for linear airfoil

Table for nonlinear airfoil

**3. Lift Coefficient to new Circulation**

$$\Gamma(y_n) = \frac{1}{2} V_\infty c_n (c_l)_n$$

Derived from Kutta-Joukowski Theorem and Lift Coefficient Definition

**4. Adjust Input**

$$\Delta\Gamma = D(\Gamma_{new} - \Gamma_{old})$$

$$\Gamma_{\text{input}} = \Gamma_{\text{old}} + \Delta\Gamma$$

# Wing Performance without Propellers using Linear Airfoil Aerodynamics

## Background

In this section, we make two simplifying assumptions. First, we have chosen to ignore the propellers. This gives us a fundamentally different problem from what we analyze. However, this gives us a good control case and will help us develop the general approach for numeric lifting lines. Second, we will apply linear aerodynamic theory. This means that we can say that $c_l = 2\pi\alpha$. We have already explored the fundamental theory behind linear aerodynamics, so we will simply apply it here to relate the angle of attack to the circulation through the coefficient.

Here, it is important to note that the fundamental difference that makes the above assumptions important is the simplifying of step 2. Since we have a linear airfoil, finding the lift coefficient becomes simpler.

## Functional Code

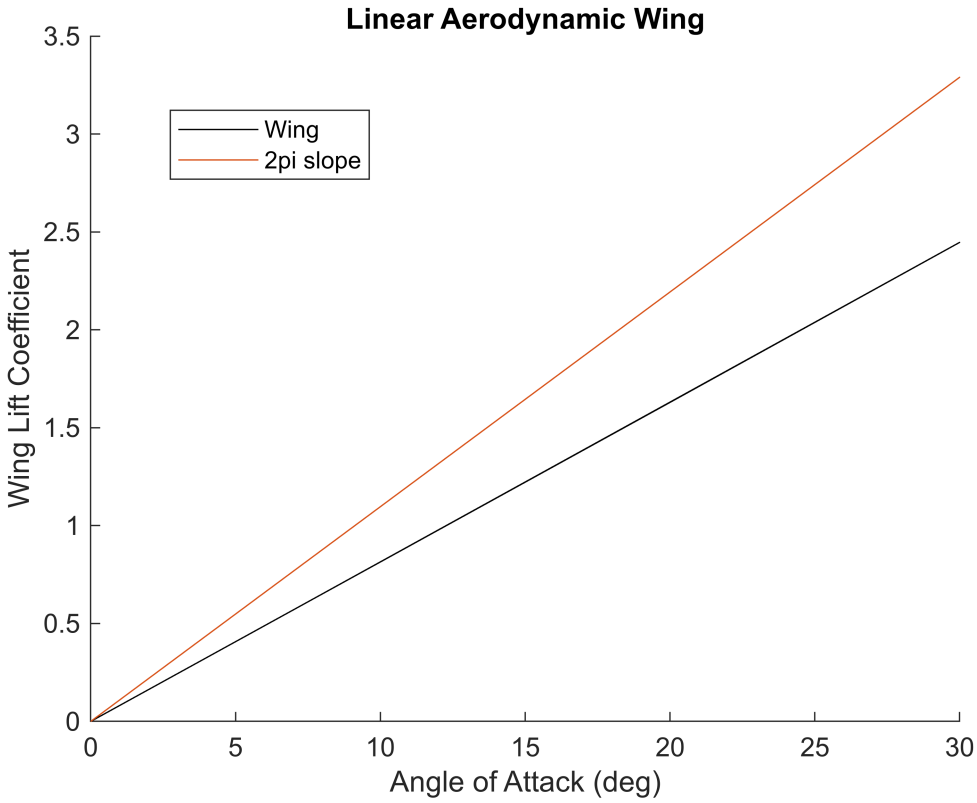
```
clear; clc;
% These are our input Parameters
L = 3.048;
N = 200;

gain = .025;
conv = .01;

chord = 0.4572;
panelLength = L/N;

ageoRange = 0:10:30;
ageoRange = [ageoRange 8];
endPoints = linspace(-L/2,L/2,N+1);
controlPoints = endPoints(1:N)+panelLength/2;



LinCl = zeros(size(ageoRange));
for i = 1:length(LinCl)
    meanError = 1;
    ageo = ageoRange(i);

    panelCircs = ellipticLift(controlPoints,L,ageo,51.44,chord);
    count = 0;
    convCount = 0;
    while(not (convCount >= 5))
        % Here we start to define the wing as a set of panels with control points
```

```
% at the middle of each panel. Each panel also has a circulation that gives
% the entire wing an initial elliptic distribution.
count = count +1;
% Here, let us go through and calculate all the trail strengths
trailVors = zeros(size(endPoints));

trailVors(1) = panelCircs(1);
for j = 2:N
    trailVors(j) = panelCircs(j)-panelCircs(j-1);
end
trailVors(end) = -panelCircs(end);

% Now that we have the trailing Strenghts, we can find the induced velocity
% at each control point.
vin = 0;
for j = 1:length(trailVors)
    denom = 4.*pi.*(controlPoints-endPoints(j));
    absAdd = trailVors(j)./denom;
    vin = vin - absAdd;
end

% Next, we induce the effective angle of attack
vinf = 51.44; %m/s

aind = atand(-vin./vinf);
aeff = ageo - aind;

% Now that we have the effective angle of attack, we ned to get the
% local cl
localCl = zeros(size(aeff));
for j = 1:length(localCl)
    localCl(j) = linearClDeg(aeff(j));
end

if(ageo == 8)
    cl8 = localCl;
end

% Now that we have the local cl value everywhere, we can calculate
% the circulation as above.
Circ_new = .5*vinf*chord*localCl;

difference = Circ_new - panelCircs;
meanError = abs(mean(difference));

if(meanError <= conv)
    convCount = convCount + 1;
end

% Finally, we update the panel circulation and restart.
```

```
        panelCircs =  panelCircs + gain*difference;
    end

    % Now that we have our circulation distribution, we need to find the
    % lift coefficients. This is done here
    lift = 1.2 * vinf * sum(panelCircs)*panelLength;
    dyPressure = .5*1.2*vinf^2;
    S = chord*L;

    LinCl(i) = lift/(dyPressure*S);

end

% Finally, we want to do our cl vs  aoa plot. It is important that we
% plotted the geometric angle of attack against lift here as that is what
% makes sense to plot for real life applications. I don't want to be out
% and about trying to design something and have to calculate the effective
% angle of attack.
% aoas = deg2rad(ageoRange);
aoaLin = ageoRange;
```

## Results

```
figure;
hold on;
plot(aoaLin,LinCl,'k-','DisplayName','Wing');
plot(aoaLin,2*pi*deg2rad(aoaLin),'DisplayName','2pi slope');
legend("Position", [0.2006,0.75484,0.17857,0.082143])
title("Linear Aerodynamic Wing")
xlabel("Angle of Attack (deg)")
ylabel("Wing Lift Coefficient")
```




### Discussion

In this section, we applied Numeric Lifting Line theory to a wing comprised of a constant cross-section of airfoils that follow linear aerodynamics. After applying the algorithm discussed above, we can get the circulation distribution in the entire wing. From here, we can find the local lift by applying the kutta-joukowski theorem. This allows us to find the total lift and therefore the lift coefficient.

Graphing the angle of attack against the lift coefficients for the wing, we can see a significant decrease in our lifting abilities. For reference, I have placed the expected $2\pi$ slope line one would expect if you were to simply apply linear aerodynamics to a full wing without accounting for induced angle of attack. Something interesting ot note here is that the lift coefficient still seems to increase linearly.

## Wing Performance without Propellers using NACA Data

### Background

The fundamental difference between this problem and the linear aerodynamics problem is how we find the lift coefficient. For this problem, instead of using an analytic formula to calculate our airfoil lift coefficients, we will be use the supplied tables instead. This allows us to have a much more complex data set for lift and also introduce drag effects into the picture.

In addition, using a table allows us to find drag properties which none of our prior tools have been able to accomplish. This allows us to then make observations on full flight characteristics which could lead to design and or purpose decisions.

## Functional Code

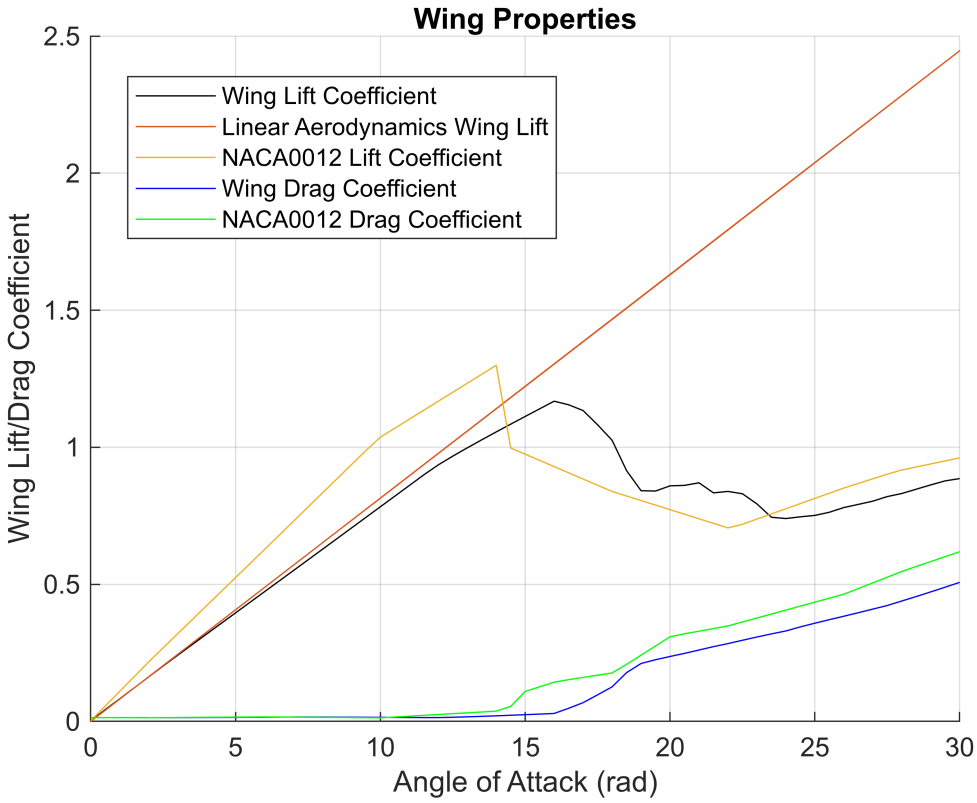
```
% These are our input Parameters
L = 3.048;
N = 200;

gain = .025;
conv = .01;

chord = 0.4572;
panelLength = L/N;

ageoRange = 0:.5:30;
endPoints = linspace(-L/2,L/2,N+1);
controlPoints = endPoints(1:N)+panelLength/2;



NACACl = zeros(size(ageoRange));
NACACd = zeros(size(ageoRange));
NACAcd8 = zeros(1,N);

for i = 1:length(NACACl)
    meanError = 1;
    ageo = ageoRange(i);

    panelCircs = ellipticLift(controlPoints,L,ageo,51.44,chord);
    count = 0;
    convCount = 0;
    while(not (convCount >= 5))
        % Here we start to define the wing as a set of panels with control points
        % at the middle of each panel. Each panel also has a circulation that gives
        % the entire wing an initial elliptic distribution.
        count = count +1;
        % Here, let us go through and calculate all the trail strengths
        trailVors = zeros(size(endPoints));

        trailVors(1) = panelCircs(1);
        for j = 2:N
            trailVors(j) = panelCircs(j)-panelCircs(j-1);
        end
        trailVors(end) = -panelCircs(end);

        % Now that we have the trailing Strenghts, we can find the induced velocity
        % at each control point.
        vin = 0;
        for j = 1:length(trailVors)
            denom = 4.*pi.*(controlPoints-endPoints(j));
            absAdd = trailVors(j)./denom;
            vin = vin - absAdd;
```

```
        end


        % Next, we induce the effective angle of attack
        vinf = 51.44; %m/s

        aind = atand(-vin./vinf);
        aeff = ageo - aind;

        % Now that we have the effective angle of attack, we ned to get the
        % local cl
        localCl = zeros(size(aeff));
        for j = 1:length(localCl)
            localCl(j) = NACAClDeg(aeff(j));
        end

        if(ageo == 8)
            NACAcl8 = localCl;
        end

        % Now that we have the local cl value everywhere, we can calculate
        % the circulation as above.
        Circ_new = .5*vinf*chord*localCl;

        difference = Circ_new - panelCircs;
        meanError = abs(mean(difference));

        if(meanError <= conv)
            convCount = convCount + 1;
        end

        % Finally, we update the panel circulation and restart.
        panelCircs =  panelCircs + gain*difference;
    end

    % Now that we have our circulation distribution, we need to find the
    % lift coefficients. This is done here
    lift = 1.2 * vinf * sum(panelCircs)*panelLength;
    dyPressure = .5*1.2*vinf^2;
    S = chord*L;

    NACACl(i) = lift/(dyPressure*S);

    % In addition to all the above, we need to go through the entire wing
    % and find the drag coefficient everywhere. We will then find the total
    % drag and the wing drag coefficient.
    drag = 0;
    cdSum = 0;
    for k = 1:length(aeff)
        cdCurr = NACACdDeg(aeff(k));
```

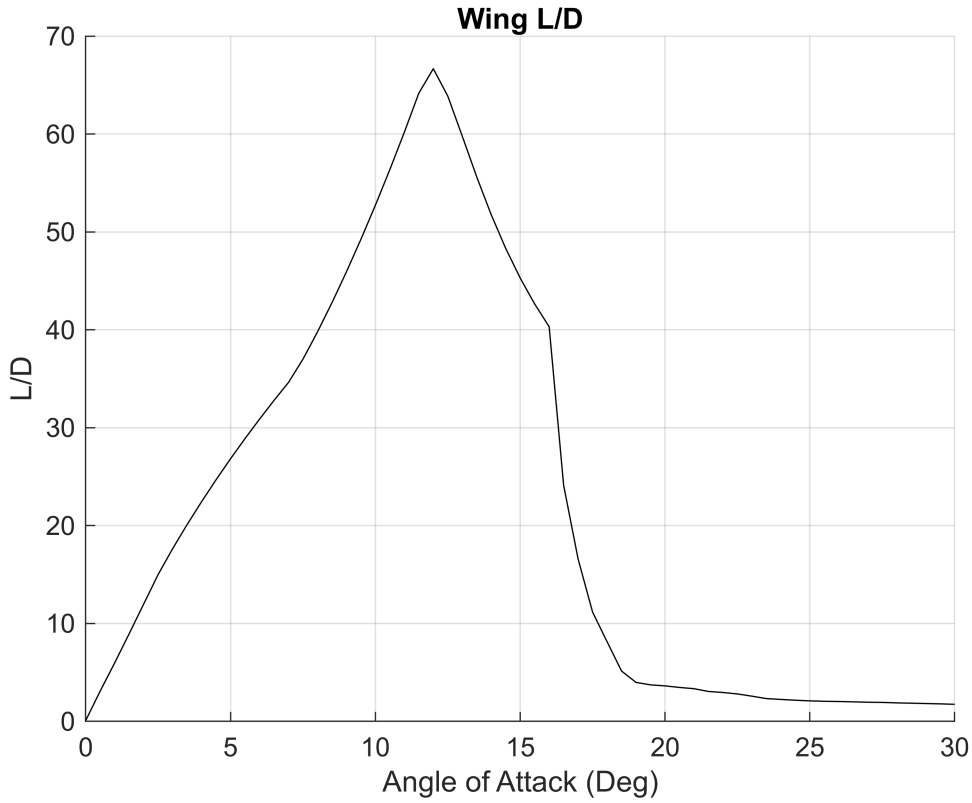
```
        cdSum = cdSum + cdCurr;
        if(ageo == 8)
            NACAcd8(k) = cdCurr;
        end
    end
    NACACd(i) = cdSum/N;



end
% Finally, we want to do our cl vs  aoa plot. It is important that we
% plotted the geometric angle of attack against lift here as that is what
% makes sense to plot for real life applications. I don't want to be out
% and about trying to design something and have to calculate the effective
% angle of attack.
% aoas = deg2rad(ageoRange);
aoaNACA = ageoRange;
LDNACA = NACACl./NACACd;

clFoil = NACAClDeg(aoaNACA);
cdFoil = NACACdDeg(aoaNACA);
```

## Results

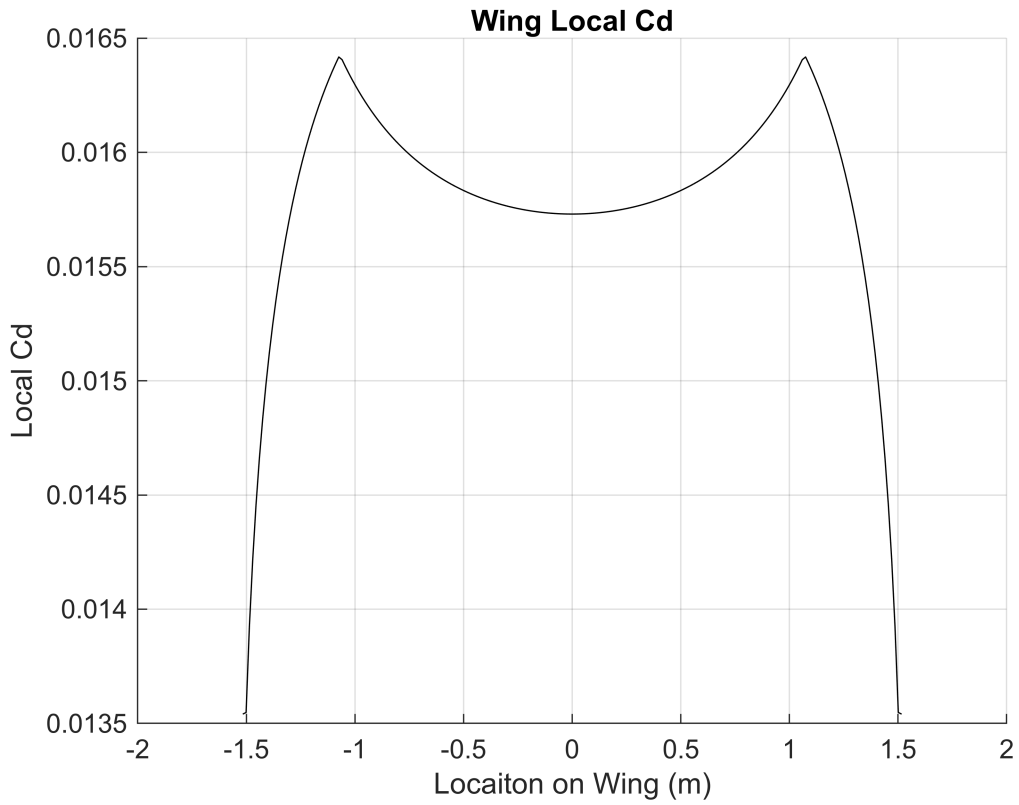
```
figure;
hold on;
plot(aoaNACA,NACACl,'k-','DisplayName','Wing Lift Coefficient');
plot(aoaLin,LinCl,'DisplayName','Linear Aerodynamics Wing Lift');
plot(aoaNACA,clFoil,'DisplayName','NACA0012 Lift Coefficient');
plot(aoaNACA,NACACd,'b-','DisplayName','Wing Drag Coefficient');
plot(aoaNACA,cdFoil,'g-','DisplayName','NACA0012 Drag Coefficient');
legend("Position", [0.1631,0.68424,0.38214,0.19286])
title("Wing Properties")
xlabel("Angle of Attack (rad)")
ylabel("Wing Lift/Drag Coefficient")
grid on
```



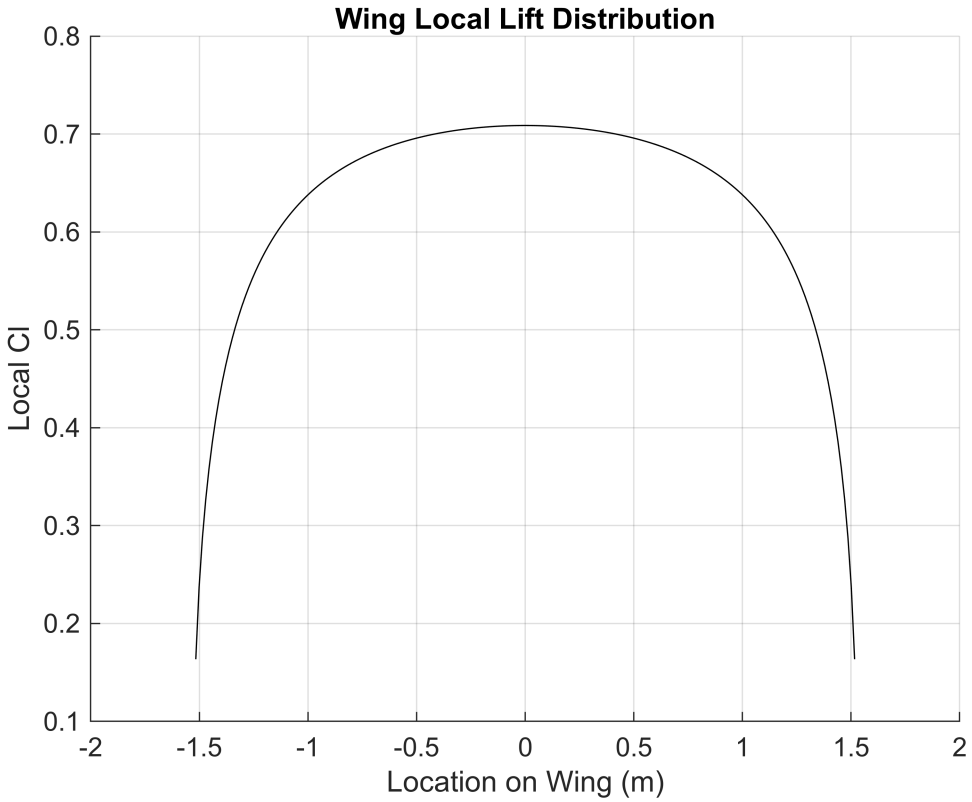

```
figure;
hold on;
plot(aoaNACA,LDNACA,'k-');

title("Wing L/D")
xlabel("Angle of Attack (Deg)")
ylabel("L/D")
grid on
```




```
figure;
hold on;
plot(controlPoints,NACAcd8,'k-');
grid on
title("Wing Local Cd")
xlabel("Locaiton on Wing (m)")
ylabel("Local Cd")
```




```
figure;
hold on;
plot(controlPoints,NACAcl8,'k-');
grid on
title("Wing Local Lift Distribution")
xlabel("Location on Wing (m)")
ylabel("Local Cl")
```




## Discussion

With this second part we have significantly more interseting things to witness.

First, let us start with the graph that shows us the lift and drag coefficients against the geometric angle of attack. This is where we can clearly see our stall angle which comes out to be approximately 16 degrees. This is actually later than the stall angle for a NACA0012 airfoil which is at 14 degrees. This makes sense as the trailing vortices lower the local angle of attack, allowing us to pitch up slightly more without stalling. However, it is important to note that our lift coefficient at stall is lower for the wing than for the airfoil.

Although we have lost lift due ot our trailing vortices, we actually profitted when it came to drag. At all angles of attack, our wing drag is noticably lower than our airfoil drag. This is a great benefit for fuel conservation, but only if we can manage the drop in drag.

We can take a moment here to recognize the predictive power of our linear aerodynamic theory. Up until around 7.5 degrees, the difference is almost imperceptible. Beyond that, our linear model stays close to the NACA model until we start approaching stall. This serves to emphasize the importance of the small angle constraint we have when working with linear airfoils.

The graph which shows us how our lift - drag ratio changes as we change our angle of attack gives us important steady state information. Since we want to analyze and make design decisions based on steady state properties at this point in the design process, it will be useful to know when we will be reaching maximum L/D efficiency. This would be the angle of attack we would like to main to maximize fuel efficiency. This happens to be at an angle of attack of approximately 12 degrees. It is interesting to note that this does not occur at the stall angle of either the wing or the airfoil but at a completely seperate angle.

The Lift Distribution graph does not show us anything extremely intresting. We can notice that the graph is not exactly elliptic, but sufficiently close that it is not worth looking at the exact shape in detail. However, the local Cd graph is much more interesting. I believe that this is likely an error in my implementation, as this shape feels almost unnatural to witness. If the domed section was flipped and more similar to the lift graph, it would be more expected. However, this is the results I am getting. Assuming I can have some degree of confidence in my implementation, this behavior is interesting to say the very least. I am not sure what would cause this to happen.

# Wing Performance with Propellers

## Background

For the final part of this analysis, we will keep the airfoil properties we analyzed in the previous section but introduce a variation in freestream velocity along the wing span. Once again, we will take fundamentally the same appraoch as the other sections, but now we will adjust the section of code which calculates the freestream velocity and induced angle of attack. This will involve reevaluating the geometric angle of attack at each chord cross-section and the magnitude of the freestream at each point. This will often entail replacing everywhere we had "vinf" with an evaluation of the freestreamWithProp function we have below. This will also impact how we calculate our lift coefficients as we will need to adjust this for the local freestream values. However, when we calculate the lift coefficient for our wing, we will use the freestream velocity without the contribution of the propellers to be consistent.

To determine the contribution of the propellar wakes, we need to determine a necessary thrust. To do this, we will assume steady operation. In this state, we need our thrust to equal our drag. We can say that our drag force is:

$$D = c_d \cdot \left(\frac{1}{2}\rho v^2\right) \cdot S = .015 \cdot 1587.64 \cdot 1.3935 = 33.18N$$

For steady operations, this will then also be our thrust force.

However, this does not account for the significant increase of drag due to the body of the aircraft. Since this section will not be as aerodynamic as the wings themselves, we will have much higher drag. To account for this, i looked up common lift to drag ratios for aircrafts and 10 seemed to be fair average. Taking this into account and noting that our lift coefficient at 8 degrees is .628, we can say that our true drag coefficnet will likely be around .063. This is roughly four times higher than what we directly calculated babove, which means we should expect a thrust of about 150 N to be in a safe operating range.

## Functional Code

```
% These are our input Parameters
L = 3.048;
N = 200;

gain = .01;
conv = .05;

chord = 0.4572;
panelLength = L/N;
Thrust = 120; % N, calculated above
```

```
ageoRange = 0:.5:30;
endPoints = linspace(-L/2,L/2,N+1);
controlPoints = endPoints(1:N)+panelLength/2;



PropCl = zeros(size(ageoRange));
PropCd = zeros(size(ageoRange));
Propcd8 = zeros(1,N);

for i = 1:length(PropCl)
    meanError = 1;
    ageo = ageoRange(i);

    panelCircs = ellipticLift(controlPoints,L,ageo,51.44,chord);
    count = 0;
    convCount = 0;

    while(not (convCount >= 5))
        % Here we start to define the wing as a set of panels with control points
        % at the middle of each panel. Each panel also has a circulation that gives
        % the entire wing an initial elliptic distribution.
        count = count +1;
%         if(rem(count,50) == 0)
%             disp(count);
%         end

        if(count >= 1000)
            disp("No convg");
            break;
        end
        % Here, let us go through and calculate all the trail strengths
        trailVors = zeros(size(endPoints));

        trailVors(1) = panelCircs(1);
        for j = 2:N
            trailVors(j) = panelCircs(j)-panelCircs(j-1);
        end
        trailVors(end) = -panelCircs(end);

        % Now that we have the trailing Strenghts, we can find the induced velocity
        % at each control point.
        vin = 0;
        for j = 1:length(trailVors)
            denom = 4.*pi.*(controlPoints-endPoints(j));
            absAdd = trailVors(j)./denom;
            vin = vin - absAdd;
        end
```

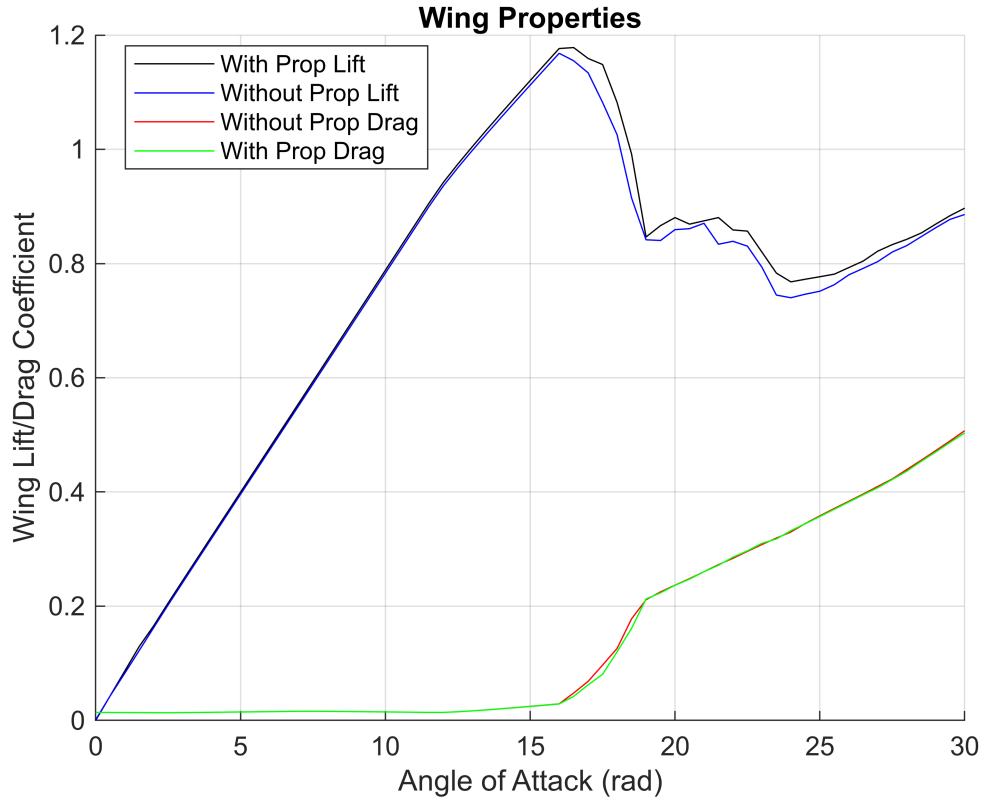
```
        % Next, we induce the effective angle of attack
        aeff = zeros(size(controlPoints));
        localCl = zeros(size(aeff));
        Circ_new = zeros(size(localCl));

        for k = 1:length(controlPoints)
            chordVec = exp(1i*-ageo);

            vinf = freestreamWithProp(controlPoints(k),Thrust,ageo);

            vinf = vinf + 1i*vin(k);

%             temp = real(vinf)*real(chordVec)+imag(vinf)*imag(chordVec);
%             temp = abs(temp);
%             temp = temp/(abs(vinf)*abs(chordVec));

            temp = (imag(vinf)/real(vinf));
            windAngle = atan(temp);
            windAngle = rad2deg(windAngle);

            aeff(k) =  windAngle+ageo;

            % Now that we have the effective angle of attack, we ned to get the
            % local cl
            localCl(k) = NACAClDeg(aeff(k));

            Circ_new(k) = .5*abs(vinf)*chord*localCl(k);
        end


%         localCl = zeros(size(aeff));
%         for j = 1:length(localCl)
%             localCl(j) = NACAClDeg(aeff(j));
%         end



        % Now that we have the local cl value everywhere, we can calculate
        % the circulation as above

        difference = Circ_new - panelCircs;
        meanError = abs(mean(difference));

        if(meanError <= conv)
            convCount = convCount + 1;
        end

        % Finally, we update the panel circulation and restart.
        panelCircs =  panelCircs + gain*difference;
    end
```

```
    % Now that we have our circulation distribution, we need to find the
    % lift coefficients. This is done here
    if(ageo == 8)
        Propcl8 = localCl;
    end
    lift = 0;
    for k = 1:length(panelCircs)
        vinf = freestreamWithProp(controlPoints(k),Thrust,ageo);
        lift = lift + 1.2 * abs(vinf) * panelCircs(k)*panelLength;
    end
    vfreestream = 51.44;
    dyPressure = .5*1.2*vfreestream^2;
    S = chord*L;

    PropCl(i) = lift/(dyPressure*S);

    % In addition to all the above, we need to go through the entire wing
    % and find the drag coefficient everywhere. We will then find the total
    % drag and the wing drag coefficient.
    drag = 0;
    cdSum = 0;
    for k = 1:length(aeff)
        cdCurr = NACACdDeg(aeff(k));

        cdSum = cdSum + cdCurr;
        if(ageo == 8)
            Propcd8(k) = cdCurr;
        end
    end
    PropCd(i) = cdSum/N;
end

% Finally, we want to do our cl vs  aoa plot. It is important that we
% plotted the geometric angle of attack against lift here as that is what
% makes sense to plot for real life applications. I don't want to be out
% and about trying to design something and have to calculate the effective
% angle of attack.
% aoas = deg2rad(ageoRange);
aoaProp = ageoRange;
LD = PropCl./PropCd;

clFoil = NACAClDeg(aoaNACA);
cdFoil = NACACdDeg(aoaNACA);
```

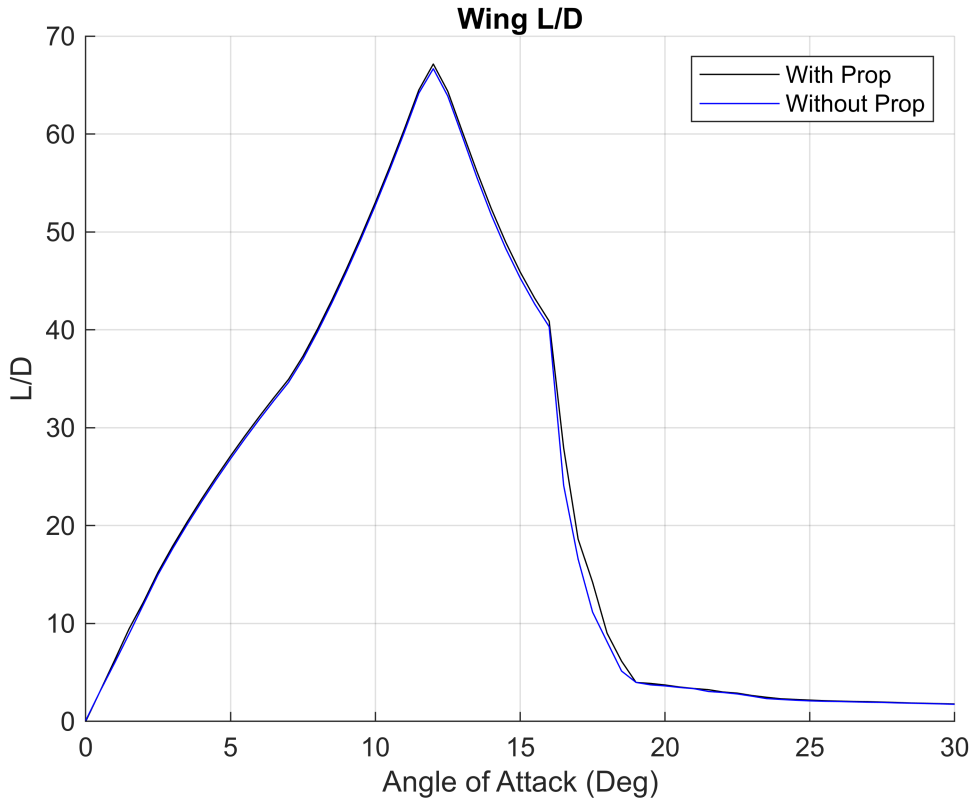
## Results

```
figure;
hold on;
plot(aoaProp,PropCl,'k-','DisplayName','With Prop Lift');
```

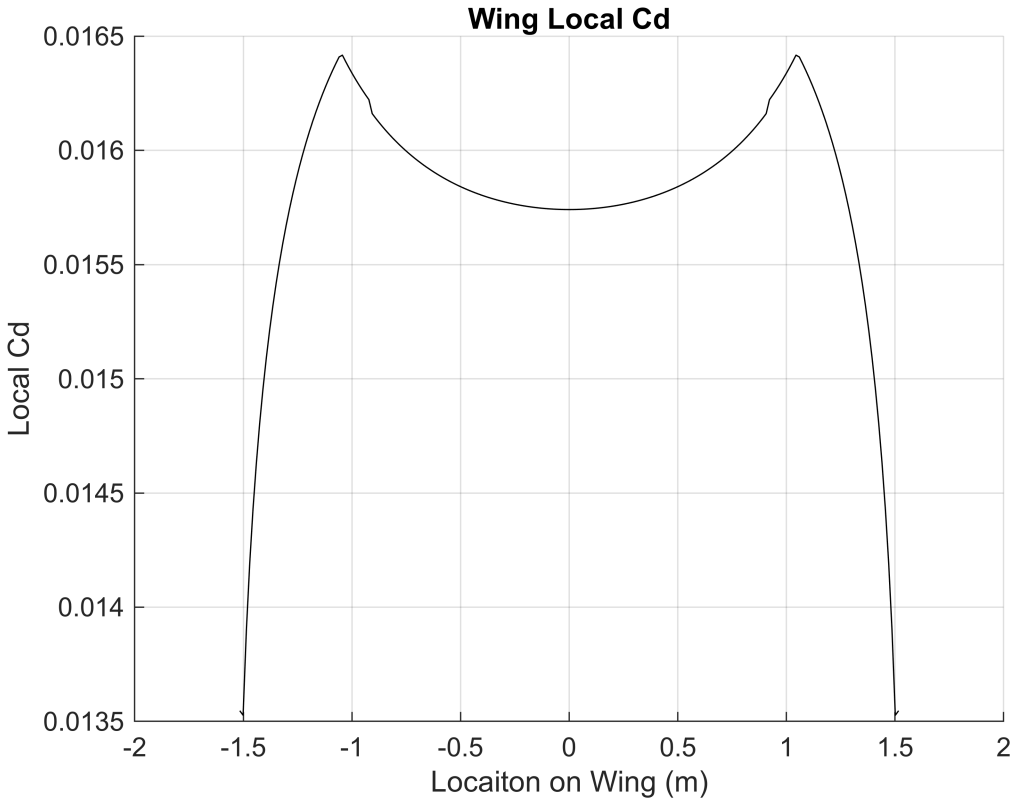
```
plot(aoaNACA,NACACl,'b-','DisplayName','Without Prop Lift');
plot(aoaNACA,NACACd,'r-','DisplayName','Without Prop Drag');
plot(aoaProp,PropCd,'g-','DisplayName','With Prop Drag');
title("Wing Properties")
xlabel("Angle of Attack (rad)")
ylabel("Wing Lift/Drag Coefficient")
grid on
legend("Position", [0.15667,0.76579,0.26964,0.14643])
```



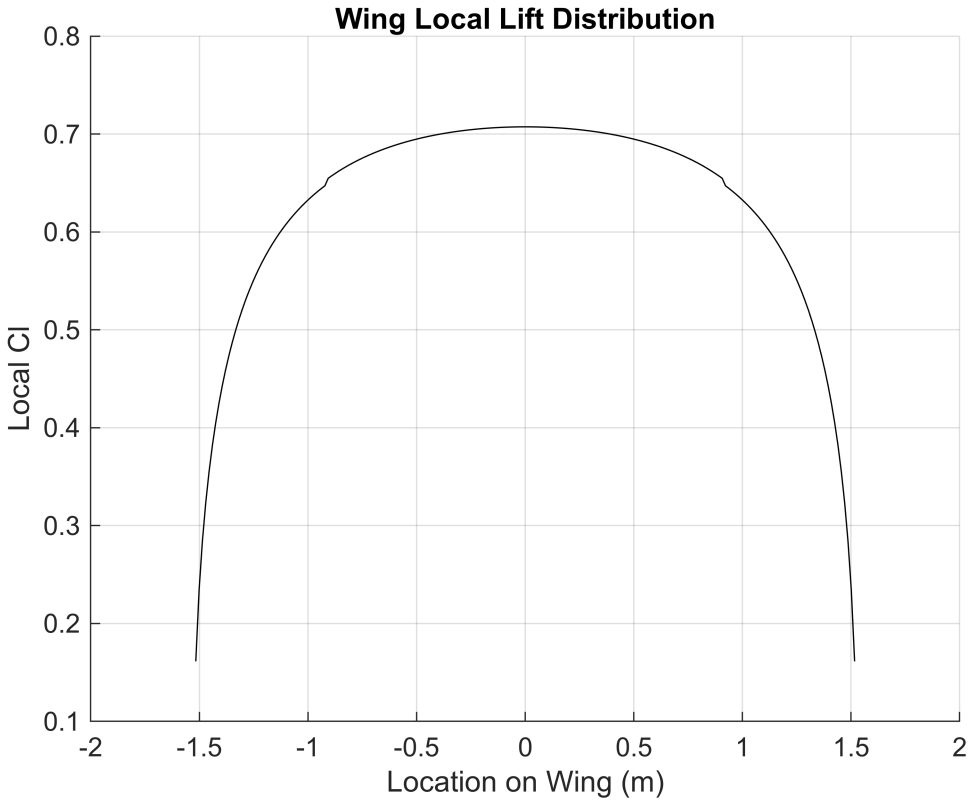

```
figure;
hold on;
plot(aoaProp,LD,'k-','DisplayName','With Prop');
plot(aoaNACA,LDNACA,'b-','DisplayName','Without Prop');
title("Wing L/D")
xlabel("Angle of Attack (Deg)")
ylabel("L/D")
grid on
legend("show")
```




```
figure;
hold on;
plot(controlPoints,Propcd8,'k-');
grid on
title("Wing Local Cd")
xlabel("Locaiton on Wing (m)")
ylabel("Local Cd")
```




```
figure;
hold on;
plot(controlPoints,Propcl8,'k-');
grid on
title("Wing Local Lift Distribution")
xlabel("Location on Wing (m)")
ylabel("Local Cl")
```




## Discussion

Once again, let us go graph by graph to analyze this section. We will be comparing the properties of our wing with propellers against the same wing without propellers as we calculated in part 2.

Drag wise, there is a marginal change to the drag coefficient of our wing. Although one could argue that there is a slight decrease in drag around 17 and 27 degrees, I would say the change is not substantial enough to make any definitive claims. On the other hand, our lift distribution has noticable changes, especially in the post stall behavior. Up to stall, lift properties of our wing is identical with or without the propellers. However, propellers give us a much kinder stall behavior. This can be seen since the propeller allows for an approximately 3 degree plateu before the lift coefficient starts to significantly decrease. This is to contrast with the no propeller wing which gives a transition period of less than .half a degree. This gives us slightly more safety when in flight since there is less risk of unexpected wind patterns inducing stall.

Notice that since our lift and drag curves were fairly close up to stall, our L/D curves look almost identical up to stall where they then start to deviate slightly. This deviation is less noticable, possibly due to the scale of the graph and scale of L/D values but still important to note. However, since our pre-stall characteristics were very similar, our opimal angle of around 12 degrees is constant in either case.

Finally, if we look at the local lift and drag curves along the wing, we will notice the interesting effect the propellers have on the wing. At the edges of the propellers, there is a noticable jump in both graphs. For our application, this jump is fairly small. However, for larger thrust propellers this jump will be more noticable and more impactful. This will likely contribute to even kinder stall characteristics. In addition, when I tested much higher thrust values to see what behavior would result, both the L/D peak and lift coefficient increase beyond the no propeller case and the drag decreased significantly below the no propeller case. This shows the potential

of Electric Aviation. It would also be interesting to see if positioning of the propellers would aid in this behavior. If having half the propeller on the wing aids, would having the entire propeller overlap with the wing help more? Is there an optimal overlap? All interesting questions, but I am ready for Christmas so it is time to break now.

Thank you for the great semester. This class was the single best class I have taken at Penn so far. It helped me learn a lot and also helped me find a potential passion. Have a lovely break!

## Helper Functions

```
function v = freestreamWithProp(y,T,aoa)
    T = T/2; % Two propellers, each with half thrust
    rho = 1.2; %kg/m3
    R = .6096; %m
    b = 3.048; %m
    vinf = complex(51.44); %m/s
    aoa = deg2rad(aoa);

    vh = sqrt(T./(2*rho*pi*R^2));
    vi = -vinf/2+sqrt((vinf/2)^2+vh^2);
    vi = vi * exp(-1i*aoa);

    if(abs(y) >= b/2-R)
        v = vinf+vi;
    else
        v = vinf;
    end
end

% This function give initial lift distribution guess we will u se.
function circ = ellipticLift(y,b,aoa,vinf,chord)
    aoa = deg2rad(aoa);
    cMax = pi*aoa*vinf*chord;
    circ = cMax*sqrt(1-(2*y/b).^2);
end

% The below functions will give me the lift coefficient for the linear and
% NACA cases
function cl = linearClRad(aoa)
    cl = 2*pi*aoa;
end

function cl = linearClDeg(aoa)
    aoa = deg2rad(aoa);
    cl = 2*pi*aoa;
end

function cl = NACAClDeg(aoa)
    sampleAlpha = [-2.025902479,.038772741,2.05982177,6.017645346,...
        9.932239783,14.16254093,14.36464087,17.98442351,...
```

```
        22.07932961,26.01754869,27.87918097,31.99647093];

    sampleCl = [-.20812679,.00652805,.22331494,.62902511,...
        1.03258364,1.30984164,1.00359647,.83948681,...
        .70332889,.85200745,.91454208,1.00542989];

    cl = interp1(sampleAlpha,sampleCl,aoa,"linear","extrap");

end

function cd = NACACdDeg(aoa)
    sampleAlpha = [-2.08998,1.992422,5.944141,10.02674,14.36712,...
        14.96876,16.05107,18.04577,20.03113,21.982,26.01005,...
        28.08664,31.939];

    sampleCd = [.014616,.013394,.016427,.013064,.039745,...
        .108421,.145077,.177656,.310901,.347753,.464323,...
        .550467,.688417];

    cd = interp1(sampleAlpha,sampleCd,aoa,"linear","extrap");

end
```