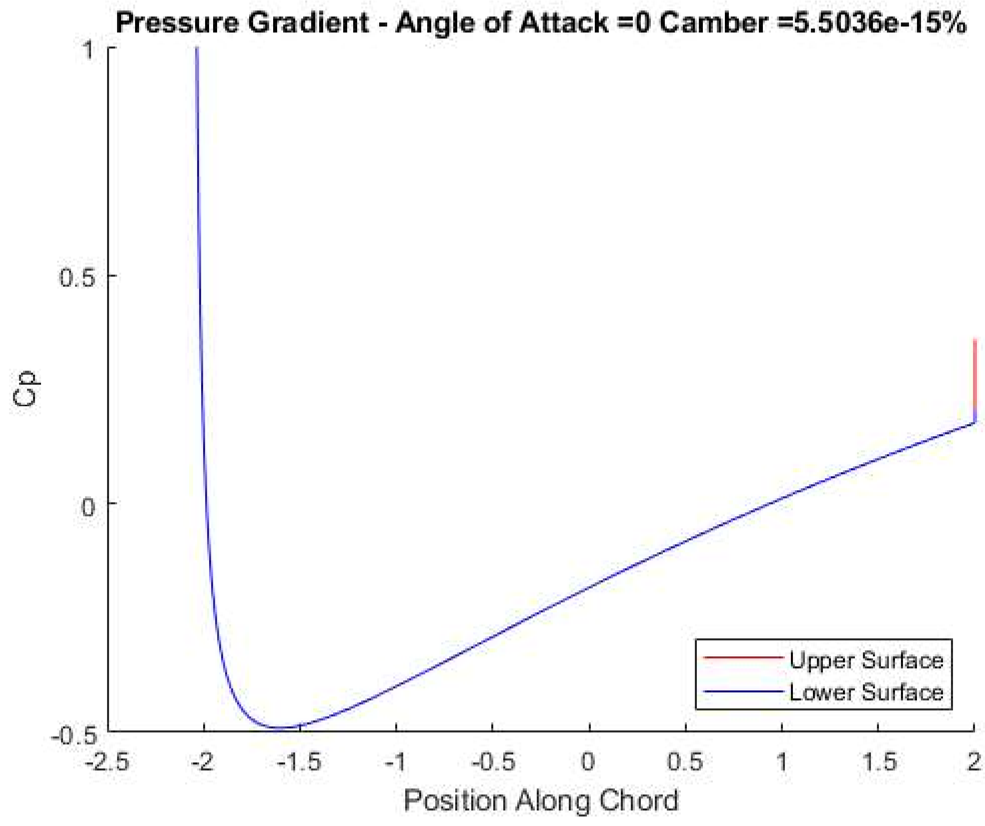
Contents

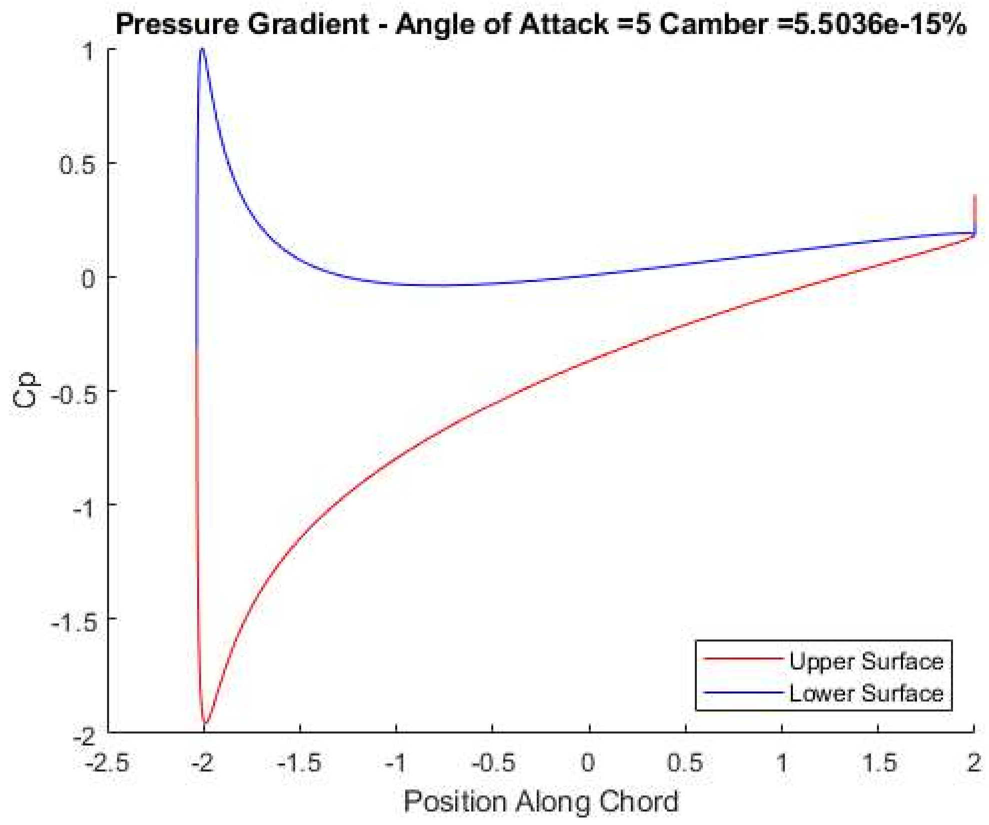
- Symmetric Airfoil
- Cambered Airfoil
- Cl-aoa Slope Comparison
- Reflection

Symmetric Airfoil

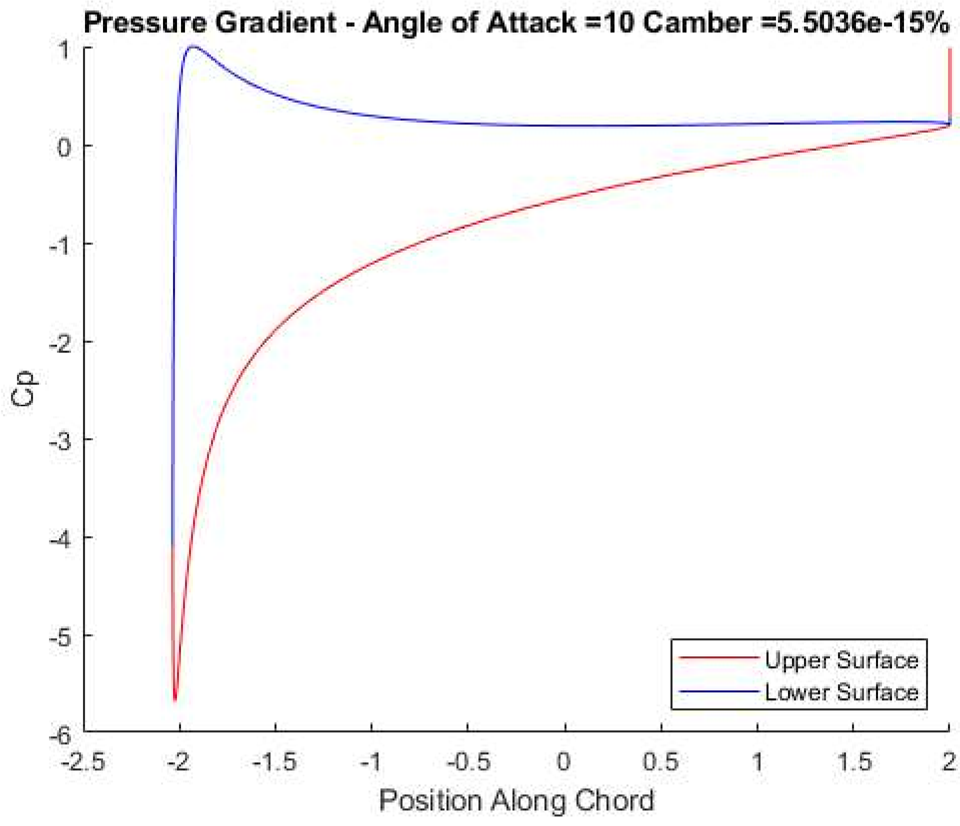
```
clear,clc;


cx1 = -1.02021;%cx = -1.02021,cy= 0 for 12% Thick Symmetric Airfoil
cy1 = 0;


J1 = Jfoil(0,cx1, cy1);
J2 = Jfoil(5,cx1, cy1);
aoa2 = deg2rad(5);
J3 = Jfoil(10,cx1, cy1);
aoa3 = deg2rad(10);
J4 = Jfoil(15,cx1, cy1);
aoa4 = deg2rad(15);

%0,-1.01363,.43961 for 12% Thick 2% Camber Airfoil
cl1 = J1.surfaceRun();
cl2 = J2.surfaceRun();
cl3 = J3.surfaceRun();
cl4 = J4.surfaceRun();

% Take average slope from each case
m1 = (cl2-cl1)/aoa2;
m2 = (cl3-cl1)/aoa3;
m3 = (cl4-cl1)/aoa4;
m4 = (cl3-cl2)/(aoa3-aoa2);
m5 = (cl4-cl2)/(aoa4-aoa2);
m6 = (cl4-cl3)/(aoa4-aoa3);

averageM = m1+m2+m3+m4+m5+m6;
averageM1 = averageM/6;

noCamSet = [cl1,cl2,cl3,cl4];
```

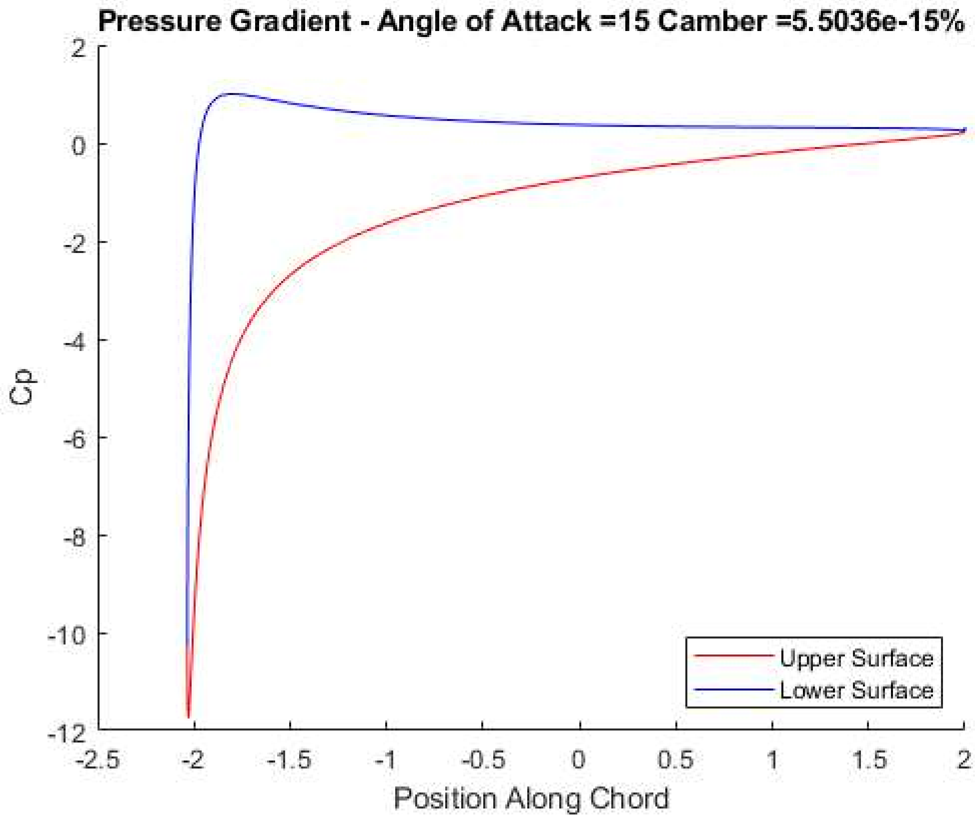


Pressure Gradient - Angle of Attack =0 Camber =5.5036e-15%
Cp
1
0.5
0
-0.5
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
Position Along Chord
Upper Surface
Lower Surface



Pressure Gradient - Angle of Attack =5 Camber =5.5036e-15%
Cp
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
Position Along Chord
Upper Surface
Lower Surface

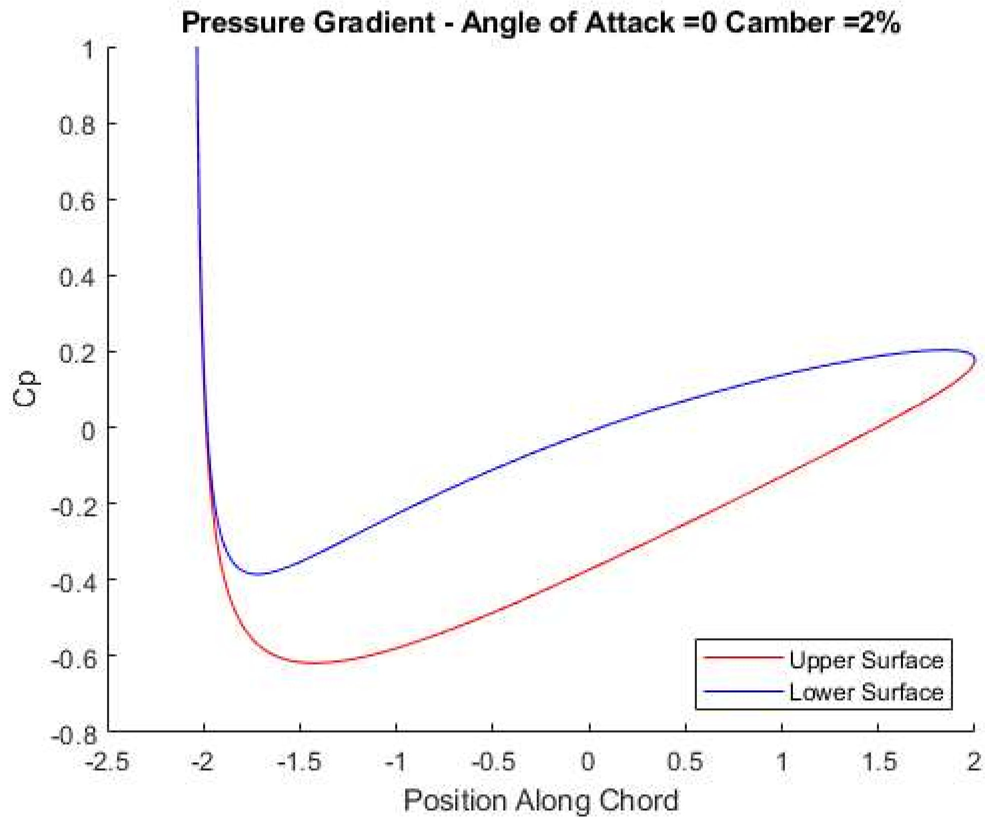




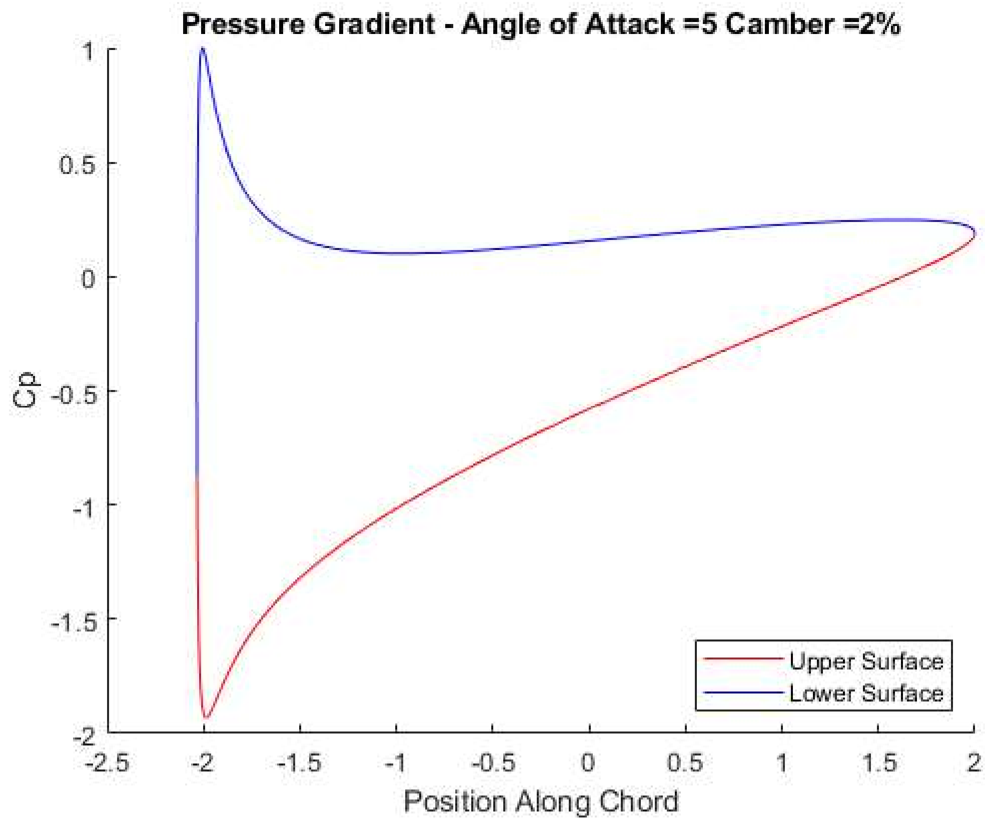
Cambered Airfoil

```
cx1 = -1.01363;%cx = -1.01363, cy= .43961 for 12% Thick 2% Camber Airfoil
cy1 = .4397;
```

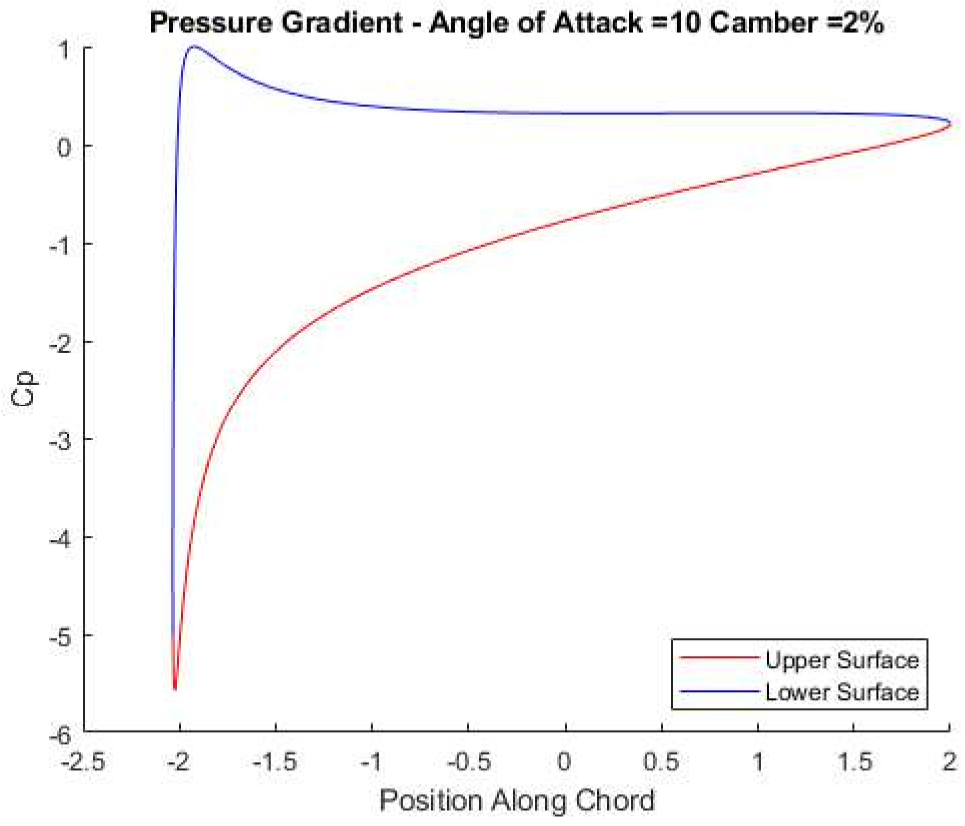
```
J5 = Jfoil(0,cx1, cy1);
aoa1 = 0;
J6 = Jfoil(5,cx1, cy1);
aoa2 = deg2rad(5);
J7 = Jfoil(10,cx1, cy1);
aoa3 = deg2rad(10);
J8 = Jfoil(15,cx1, cy1);
aoa4 = deg2rad(15);


cl1 = J5.surfaceRun();
cl2 = J6.surfaceRun();
cl3 = J7.surfaceRun();
cl4 = J8.surfaceRun();

% Take average slope from each case
m1 = (cl2-cl1)/aoa2;
m2 = (cl3-cl1)/aoa3;
m3 = (cl4-cl1)/aoa4;
m4 = (cl3-cl2)/(aoa3-aoa2);
m5 = (cl4-cl2)/(aoa4-aoa2);
m6 = (cl4-cl3)/(aoa4-aoa3);

averageM = m1+m2+m3+m4+m5+m6;
averageM2 = averageM/6;

camSet = [cl1,cl2,cl3,cl4];
```

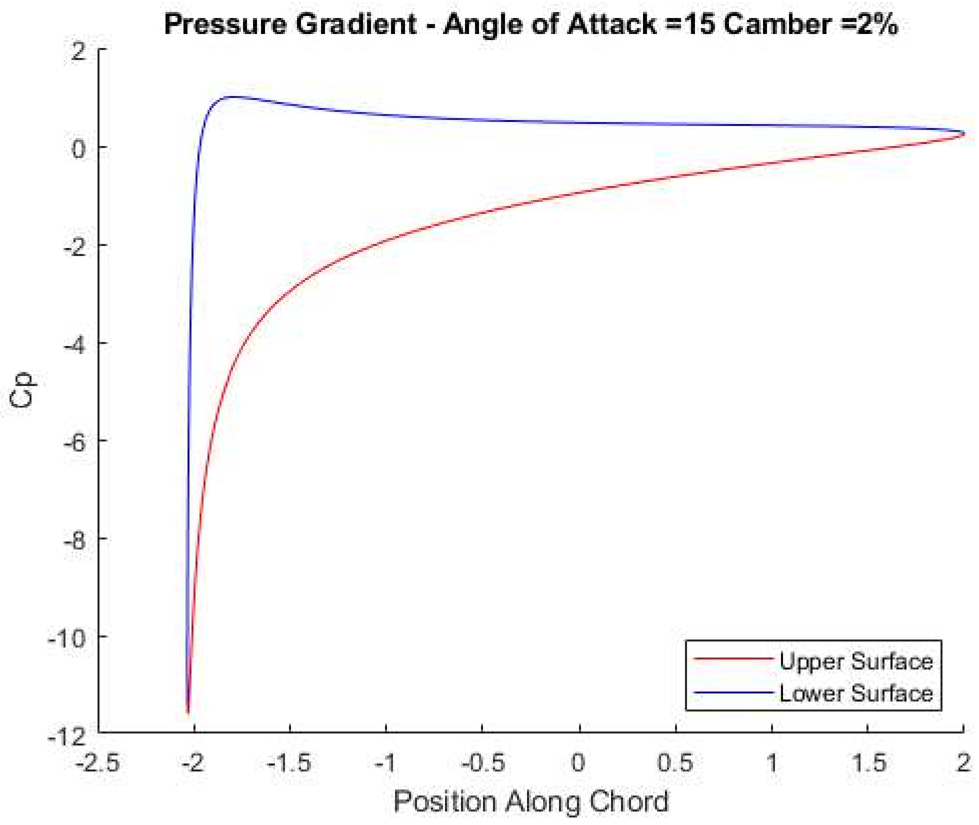


Pressure Gradient - Angle of Attack =0 Camber =2%
Cp
Position Along Chord
Upper Surface
Lower Surface
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2



Pressure Gradient - Angle of Attack =5 Camber =2%
Cp
Position Along Chord
Upper Surface
Lower Surface
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

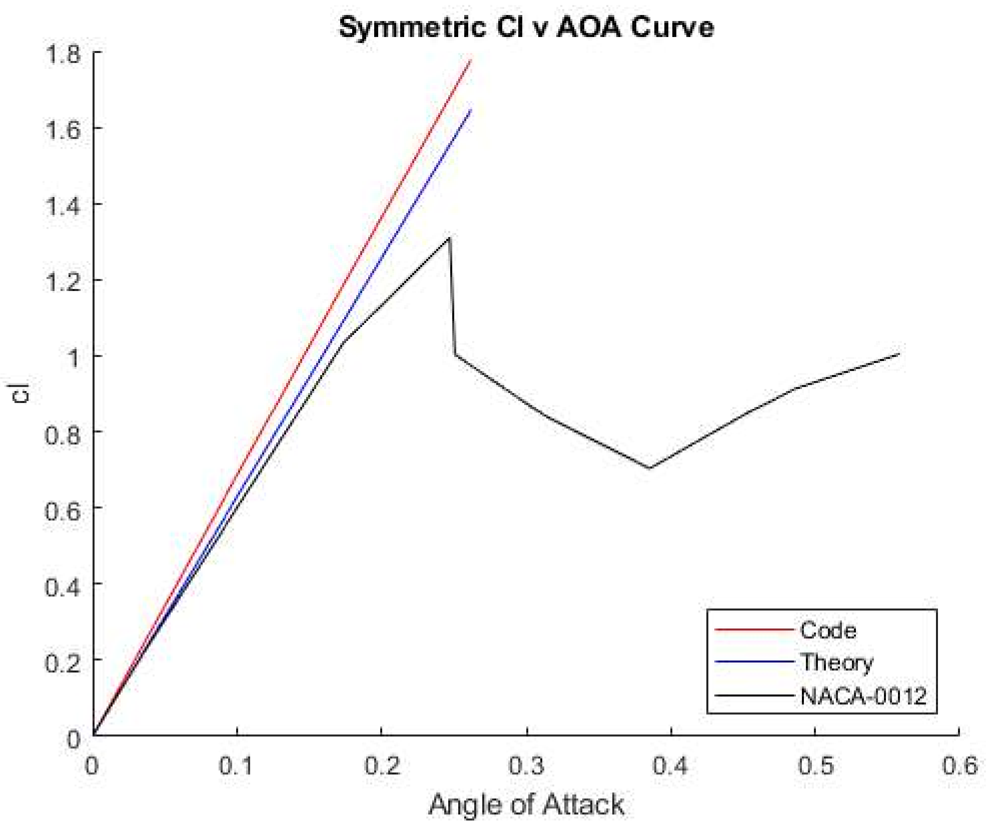


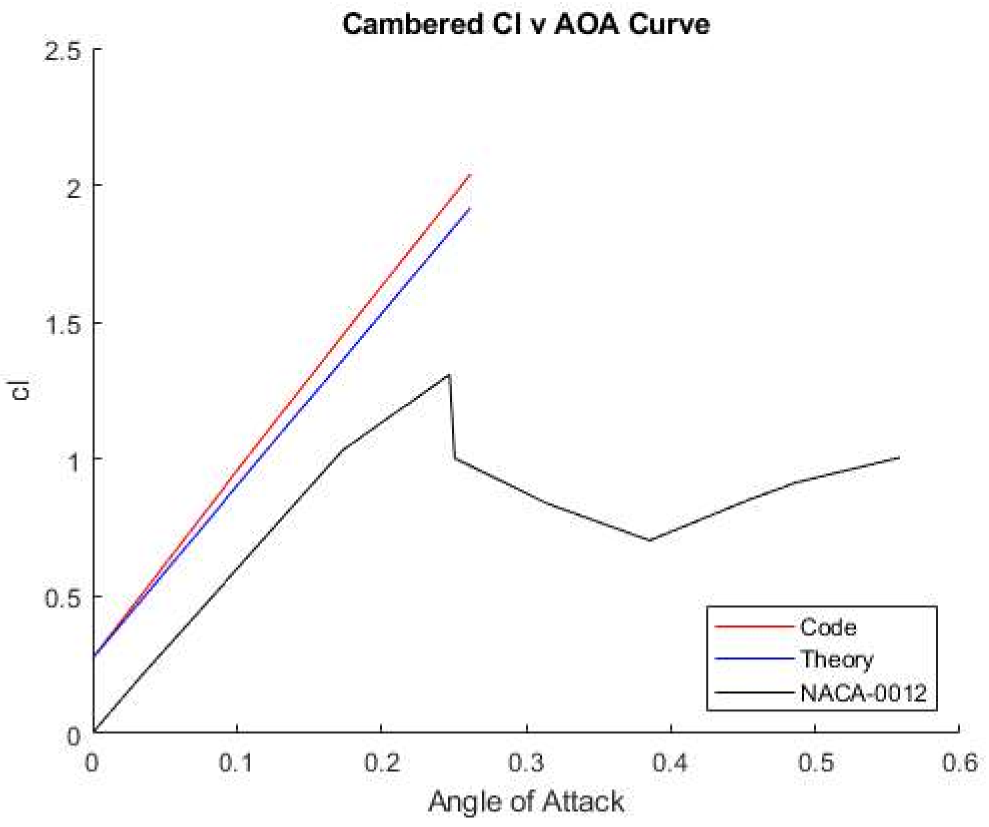


Cl-aoa Slope Comparison

```
%Here,we will plot the lift coefficient to the angle of attack. We also
%plotted the theory curve in each case to see our variation from each.
xset = linspace(0,15,4);
xset = deg2rad(xset);
yset = 2*pi*xset;

figure;
NACAaoa = [0.038772741,2.05982177,6.01864536,9.932239783,14.16254093,...
    14.36464087, 17.98442351,22.07932961,26.01753869,27.87918097,...
    31.99647093];
NACAaoa = deg2rad(NACAaoa);
NACAcl = [0.00652805,.22331494,.62902511,1.03258364,1.30984164,...
    1.00359647,.83948681,.70332889,.85200745,.91454208,...
    1.00542989];
hold on;
title('Symmetric Cl v AOA Curve');
plot(xset,noCamSet,'r-');
plot(xset,yset,'b-');
plot(NACAaoa,NACAcl,'k-')
xlabel('Angle of Attack');
ylabel('cl');
legend('Code','Theory','NACA-0012', 'Location','southeast');

yset = yset+cl1;
figure;
hold on;
title('Cambered Cl v AOA Curve');
plot(xset,camSet,'r-');
plot(xset,yset,'b-');
plot(NACAaoa,NACAcl,'k-')
xlabel('Angle of Attack');
ylabel('cl');
legend('Code','Theory','NACA-0012', 'Location','southeast');


disp('Slope of Cl-aoa line, 0% Camber');
disp(averageM1);

disp('Slope of Cl-aoa line, 2% Camber');
disp(averageM2);
```

```
Slope of Cl-aoa line, 0% Camber
    6.7882

Slope of Cl-aoa line, 2% Camber
    6.7490
```





Reflection

At this point, we can see some of the power that comes with the joukowski transform. The results given were fairly accurate. Although we did not see exactly a slope of 2*pi, our results were very close to this value. The error here likely results from several different sources. The first that somes to my mind is viscous effects. Next, we should notice that we made small angle approximations in our derivation of some of the theorems about. Additonal, we should notice that Joukowsky airfoils are also not perfectly physical as a cusp forms at the trailing edge. Additionally, our potentially flow assumptions add another layer of error. Together, these effects work to change our lift, more often dissipating lift than agumenting it. As such, our slope discrepency of .5 is likely due to these effects.

```
% A effect of notice here is the little spike at the end of symmetric
% graphs. I believe that this is a result of the kutta condition creating a
% stagnation point at the trailing edge. However, since we had to
% discritize our system the Cp 1 zone near these occurs very quickly. I
% believe if we zoom in on that area and refine the mesh, we would see a
% smooth but steep transition.
```

Contents

- Geometry Creation
- Flow Parameters
- Graphs
- Helper Functions

```
classdef Jfoil
   properties
       angle
       cx
       cy
   end

   methods
       function obj = Jfoil(aoa,cx,cy) %Constructor
           %disp(aoa);
           obj.angle = aoa;
           obj.cx = cx;
           obj.cy = cy;
       end

       function cl = surfaceRun(obj)
```

Geometry Creation

This makes the Circle

```
             N = 1000;
             scaleFactor = 10;
             Cx = obj.cx;
             Cy = obj.cy;

             Cx = Cx/scaleFactor;
             Cy = Cy/scaleFactor;
             Center = Cx + Cy*1i;
             P1 = complex(1);

             R = abs(Center - P1);

             thetas = linspace(0,2*pi,N);

             circle = Center + R*exp(1i.*thetas);

             %This Makes the Airfoil
             airfoil = jMap(circle);
             P2 = complex(1);
             % Find maximum y value, double to find thickness since we have symmetric
             % airfoil:
             maxX = 0;
             minX = 0;

             thickness = 0;
             camber = 0;

              for(k = 1:numel(airfoil))
```

```
%                 if(imag(airfoil(k))>maxY)
%                     maxY = imag(airfoil(k));
%                 end
%
%                 if(imag(airfoil(k))<minY)
%                     minY = imag(airfoil(k));
%                 end
%
                thick = imag(airfoil(k)) - imag(airfoil(N-k+1));

                if(thick > thickness)
                    thickness = thick;
                end

                camb = (imag(airfoil(k))+imag(airfoil(N-k+1)))/2;
                if(camb > camber)
                    camber = camb;
                end

                if(real(airfoil(k))>maxX)
                    maxX = real(airfoil(k));
                end

                if(real(airfoil(k))<minX)
                    minX = real(airfoil(k));
                end
             end

            %This is to make sure we are answering the real question
            chordLength = maxX-minX;
            %These are used just to make sure airfoil is valid
            thickPercent = thickness*100/chordLength;
            cambPercent = camber*100/chordLength;
```

Flow Parameters

```
            V = 10;

            aoa = obj.angle; % Degress
            aoa = deg2rad(aoa);

            Circ = 4*pi*V*R*sin(aoa+asin(Cy/R));

             position = circle;

             temp1 = V*exp(-1i*aoa);
             temp2 = 1i*Circ./(2*pi*(position - Center));
             temp3num = -V*R*R*exp(1i*aoa);
             temp3denom = (position - Center).*(position-Center);
             temp3 = temp3num./temp3denom;

             velocity = temp1 + temp2 + temp3;
             velocity = conj(velocity);
```

Graphs

First some formatting and pre-processing

```
%figure;
%hold on;
%axis([-5 5 -5 5]);
[airMap,airVel] = jMapVel(position,velocity);
velocityMags = abs(airVel);

upperSurface = airMap(1:round(N/2));
upperVels = velocityMags(1:round(N/2));

lowerSurface = airMap(round(N/2):N);
lowerVels = velocityMags(round(N/2):N);


%Visual plot of Airfoil
%plot(upperSurface,'r--');
%plot(lowerSurface,'b--');

figure;
aoastr = num2str(rad2deg(aoa));
camstr = num2str(cambPercent);
tit = strcat('Pressure Gradient - Angle of Attack = ',aoastr,' Camber = ',camstr,'%');
title(tit);
xlabel('Position Along Chord')
ylabel('Cp');

hold on;
CpUpper = 1-(upperVels/V).^2;
CpLower = 1- (lowerVels/V).^2;


plot(real(upperSurface),CpUpper,'r-');

plot(real(lowerSurface),CpLower,'b-');

legend('Upper Surface','Lower Surface', 'Location','southeast');

% Kutta Joukowsky Theorem to calculate lift coefficient.
kuttaLift = 1.2*V*Circ;
q = .5*1.2*V^2;
cl = kuttaLift/(q*chordLength);
```

Helper Functions

```
function w = jMap(z)
    w = z+1./z;
end

function [w,airVel] = jMapVel(z,v)
    w = z+1./z;

    v = conj(v);
    denom = 1-1./(z.*z);
    ret = v./denom;

    airVel = conj(ret);
end
```

```
            end
        end
    end
```

Published with MATLAB® R2020a