AEE 342: Aerodynamics, Project 1b – Analysis of Symmetric Airfoils

Submitted: 01/30/15

In the study of fluid flows, it is of vital importance to perform detailed and thoughtful analysis of a fluids behavior as it interacts with an object. Such analyses may be simulated through mathematical modeling with computational tools. However, these tools are of little use if mathematical models cannot be constructed to reflect situations and interactions likely to be encountered in real life. That being the case, the problem under investigation demands precisely such modelling of a NACA 0015 airfoil and the subsequent analysis of the flow behavior around it. Fundamental to the generation of these flow conditions is the notion of a source or a sink. These terms, respectively, refer to points with either a very high velocity potential or a very low one. It is this way that sources and sinks appear to 'deflect' or 'attract' local flow depending on their strengths. With careful manipulation of these points, their positions, and their strengths, an increasingly precise representation of flow past an airfoil may be modelled. It is the convergence of this representation on expected, experimental results that lies at the crux of this investigation.

The flow being studied is defined by a flowfield given by the general system

$$u(x,y) = 1 + \sum_{i=1}^{n} s_i \frac{x - x_i}{(x - x_i)^2 + (y - y_i)^2}$$

$$v(x,y) = 1 + \sum_{i=1}^{n} s_i \frac{y - y_i}{(x - x_i)^2 + (y - y_i)^2}$$

where u is the x component of the velocity field, and v is the y component. This the general form of a 2 dimensional velocity field with n sources/sinks advancing with iterator i. x and y are coordinates of a point for which the velocity is being evaluated, while x_i and y_i are the coordinates of a particular source/sink. These locations are placed appropriately on the x-axis (for a symmetric airfoil) and between 0 and 1 (for the convenience of having unit chord) by defining

$$x_i = \frac{i}{n+1}$$

The relevant area is centered by defining the bounds of the velocity field as

$$-1 \leq x \leq 2$$

$$-1 \leq y \leq 1$$

An airfoil is defined for analysis, the equation of which is given by the NACA thickness distribution equation

$$y = \frac{t}{0.20}(0.2969\sqrt{x} - 0.1260x - 0.3516x^2 + 0.2843x^3 - 0.1015x^4)$$

where thickness t for a NACA 0015 airfoil is $t = 0.15$. This is used to generate the airfoil's surface. In order to determine the strengths of the sources/sinks, the stream function $\psi(x_j, y_j)$ must be set equal to zero at the airfoil's surface. This ensures that there is no flow through the airfoil boundary. $\psi(x_j, y_j)$ can be found by integrating the equations of the velocity field, yielding

$$\tan^{-1}\frac{y_j}{x_j - x_i}s_i + y = 0$$

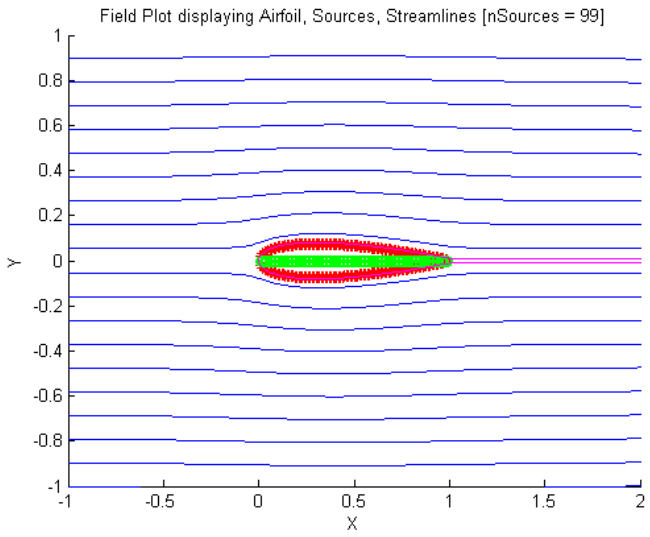
By creating a linear system containing such equations for each source/sink, the value of each source/sink may be found. With this information, the flow behavior around the NACA 0015 airfoil can successfully be modelled. The behavior of this airfoil can then be predicted by computing the pressure coefficient distribution along its chord, which is given by

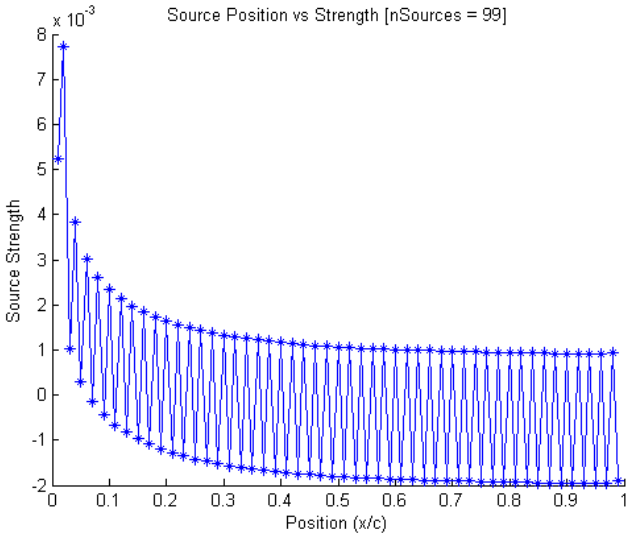
$$C_p \equiv \frac{p - p_\infty}{\frac{1}{2}\rho_\infty {V_\infty}^2} = 1 - \frac{u^2 + v^2}{{V_\infty}^2}$$

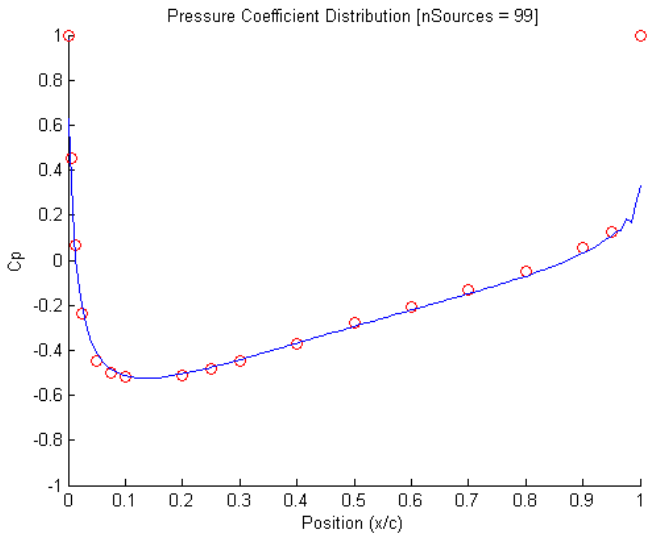
where p, ρ, V are pressure, density, and velocity, respectively. The value of C_p is evaluated at each x value along the airfoil in order to find a distribution in order to compare to NACA experimental table values. The deviation of the simulated airfoil's behavior from that of the real airfoil is indicative of how successful the model is and whether additional refinement is necessary. This deviation is determined by calculating the root mean square error of the calculated values of C_p against values interpolated from a table of NACA values. This error is given by

$$RMS = \sqrt{\frac{\sum_{j=1}^{n}\left(C_{p,computed,j} - C_{p,data,j}\right)^2}{n}}$$

Finally, all that remains is to repeat calculations for a range of values of the number is sources/sinks and determine the error associated with each. Namely, it is critical to determine whether the solution converges with more sources/sinks. Then, depending on the requirements of the specific problem, the accuracy of the simulation can be set within a reasonable range by following this convergence.





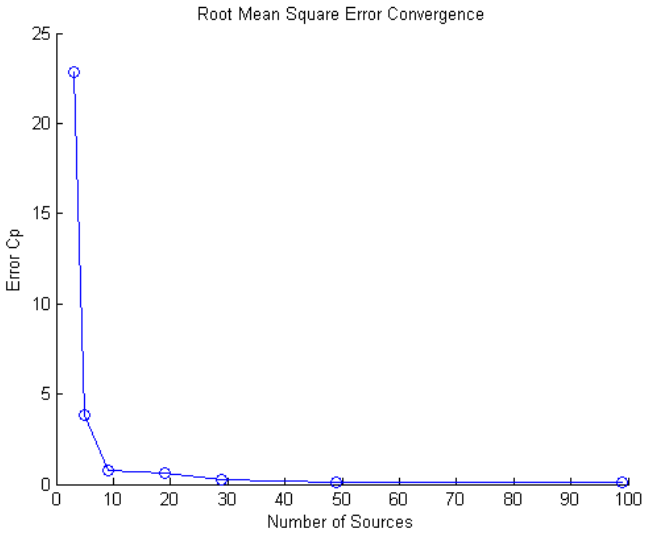


The main plot generated in this investigation is the general field plot. This gives the best overview of the flow being modelled, showing the surface of the airfoil, source/sink locations, streamlines, and streamlines at the surface of the airfoil. It is clear here that with 99 sources/sinks, the airfoil appears as it should and behaves as expected, at least on a very superficial level. Further investigation and other visualizations of the flow and airfoil performance will yield more conclusive results.

The next plot to consider is of the distribution and strengths of the sources/sinks plotted. Immediately it is clear that the greatest source strength lies on the leading edge of the airfoil. This is consistent with observations of the plot of the airfoil itself, because it is thickest near the leading edge, and because the greatest potential is required to divert the flow when it is coming from the relatively great velocity of the freestream. Approaching the trailing edge, the source/sink strength appears to level out, likely because little work is needed to influence the flow here. Sources/sinks in this region likely contribute mostly to refining the airfoil's shape.

The third plot shown depicts the distribution of the pressure coefficient in the simulation involving 99 sources/sinks. The curve of the distribution is very smooth and

clearly lies very close to the table values given by circles. This suggests that the error in this simulation is very low, but the following plot will offer more insight into the convergence of the model with physical principles.



The fourth and final plot shows the convergence of the root mean square error as the number of sources/sinks is increased. It is clear that with few sources/sinks, the error is very high; unacceptable for many applications. However, adding more sources/sinks leads to a rapid drop in error, soon bringing it within a reasonable range. The error associated with 99 sources/sinks is clearly shown to be very low, and the overall trend suggests that more sources/sinks may be added to satisfy any particular error threshold. However, the curve also appears asymptotic, suggesting that reduction in error will diminish as more and more sources/sinks are placed. For this reason, it is critical that the problem be well defined so that appropriate compromises can be made.

Calculations

```
%Joel Lubinitsky
%AEE 342 - Project 1b: Analysis of Symmetric Airfoils
%01/30/15

clear all
close all
clc

%% Known
%Domain
xMin = -1;
xMax = 2;

yMin = -1;
yMax = 1;

%NACA0015
t = 0.15;

%Pressure Coefficients (NACA0015)
ratioPositionChord = [0 0.005 0.0125 0.025 0.050 0.075 0.10 0.20 0.25 0.30 :
0.1 : 0.90 0.95 1.00]';
coefficientPressureExp = [1.000 0.454 0.067 -0.237 -0.450 -0.498 -0.520 -0.510
-0.484 -0.450 -0.369 -0.279 -0.206 -0.132 -0.049 0.055 0.128 1.000]';

%% Calculations
nSinks = 99;

xSink = zeros(1, nSinks);
for i = [1 : length(xSink)]
    xSink(i) = i / (nSinks + 1);
end

yAirfoil = (t ./ 0.20) .* (0.2969 .* sqrt(xSink) - 0.1260 .* xSink - 0.3516 .*
xSink .^ 2 + 0.2843 .* xSink .^ 3 - 0.1015 .* xSink .^ 4);

M = zeros(nSinks, nSinks);
for j = [1 : nSinks]
    for i = [1 : nSinks]
        M(j, i) = atan2(yAirfoil(j), (xSink(j) - xSink(i)));
    end
end

R = -yAirfoil';
s = M\R;

[x, y] = meshgrid(linspace(xMin, xMax, 30), linspace(yMin, yMax, 20));

u = 1;
v = 0;
for i = [1 : nSinks]
    u = u + s(i) .* (x - xSink(i)) ./ ((x - xSink(i)) .^ 2 + y .^ 2);
```

```
    v = v + s(i) .* y ./ ((x - xSink(i)) .^ 2 + y .^ 2);
end

%Initialize Loop
T = 10;
dt = 0.01;
N = (T / dt) + 1;
xy = zeros(N, 2);

%Run Loop
figure(1)
hold on
title('Field Plot displaying Airfoil, Sources, Streamlines [nSources = 99]')
xlabel('X')
ylabel('Y')
axis([xMin xMax yMin yMax])

for i = [1 : 20]
    xy(1, :) = [x(1), y(i)];
    for n = [1 : N - 1]
        xy(n + 1, :) = p1bEuler(xy(n, :), s, xSink, dt);
    end
    plot(xy(:, 1), xy(:, 2))
end

plot(xSink, yAirfoil, '*', 'color', [1 0 0])
plot(xSink, -yAirfoil, '*', 'color', [1 0 0])
plot(xSink, 0, 'o', 'color', [0 1 0])

%Airfoil Streamlines
xyAirfoil = zeros(N, 2);
for i = [-0.001, 0.001]
    xyAirfoil(1, :) = [0, i];

    for n = [1 : N - 1]
       xyAirfoil(n + 1, :) = p1bEuler(xyAirfoil(n, :), s, xSink, dt);
    end

    plot(xyAirfoil(:, 1), xyAirfoil(:, 2), 'color', [1 0 1])
end

%Pressure Coefficients
velocityFreestream = sqrt(mean(u(:, 1)) .^ 2 + mean(v(:, 1)) .^ 2);
[minEndAirfoil, indexEndAirfoil] = min(abs(xyAirfoil(:, 1) - 1));

uAirfoil = 1;
vAirfoil = 0;
for i = [1 : nSinks]
    uAirfoil = uAirfoil + s(i) .* (xyAirfoil(:, 1) - xSink(i)) ./
((xyAirfoil(:, 1)- xSink(i)) .^ 2 + xyAirfoil(:, 2).^ 2);
    vAirfoil = vAirfoil + s(i) .* xyAirfoil(:, 2)./ ((xyAirfoil(:, 1) -
xSink(i)) .^ 2 + xyAirfoil(:, 2).^ 2);
end

qAirfoil = sqrt(uAirfoil .^ 2 + vAirfoil .^ 2);
```

```
coefficientPressureSim = 1 - (qAirfoil .^ 2) ./ (velocityFreestream .^ 2);

%Root Mean Square Error
nSinkValues = [3 5 9 19 29 49 99];
errorRMS = zeros(length(nSinkValues), 1);
for n = [1 : length(nSinkValues)]
    errorRMS(n) = p1bErrorRMS(nSinkValues(n));
end

%% Plots
%Source-Sink Distribution
figure(2)
hold on
title('Source Position vs Strength [nSources = 99]')
xlabel('Position (x/c)')
ylabel('Source Strength')
plot(xSink, s, '-*')

%Pressure Coefficients
figure(3)
hold on
axis([0 1 -1 1])
title('Pressure Coefficient Distribution [nSources = 99]')
xlabel('Position (x/c)')
ylabel('Cp')
plot(ratioPositionChord, coefficientPressureExp, 'o', 'color', [1 0 0])
plot(xyAirfoil(:, 1), coefficientPressureSim)

%RMS Error
figure(4)
hold on
title('Root Mean Square Error Convergence')
xlabel('Number of Sources')
ylabel('Error Cp')
loglog(nSinkValues, errorRMS, '-o')
```

Functions

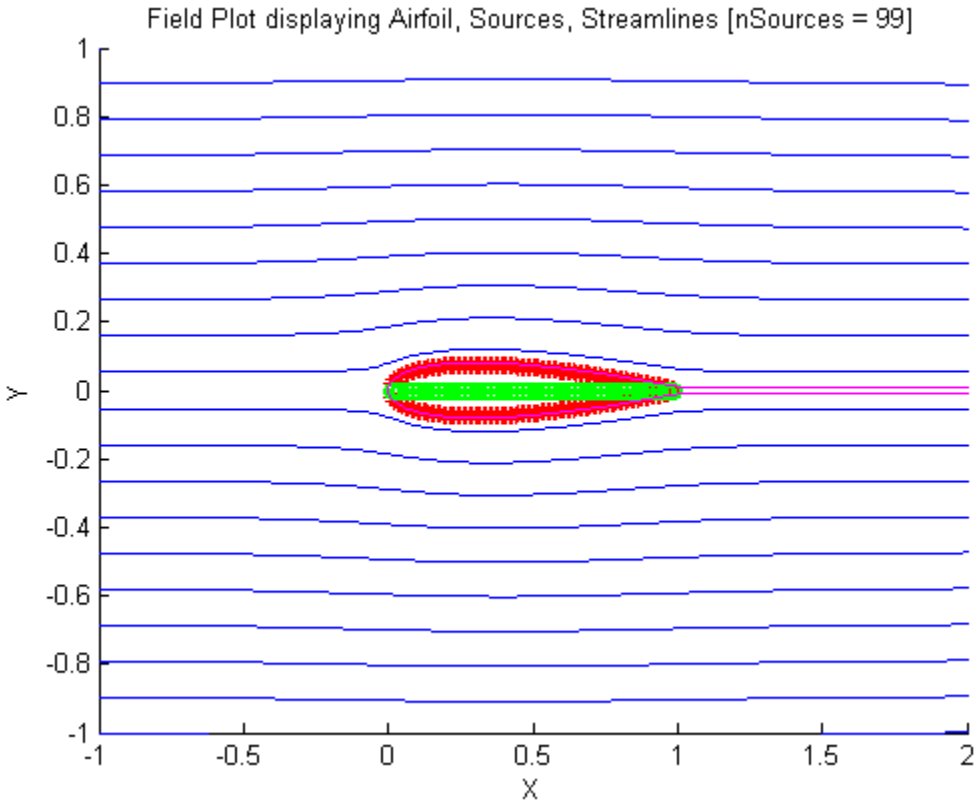
```
%Joel Lubinitsky
%AEE 342 - Project 1a: Analysis of Symmetric Airfoils
%Euler Loop Function
%01/30/15

function xyNext = p1bEuler(xy, s, xSink, dt)
xyNext = zeros(1, 2);

x = xy(1);
y = xy(2);

u = 1;
v = 0;
for i = [1 : length(s)]
    u = u + s(i) .* (x - xSink(i)) ./ ((x - xSink(i)) .^ 2 + y .^ 2);
    v = v + s(i) .* y ./ ((x - xSink(i)) .^ 2 + y .^ 2);
```

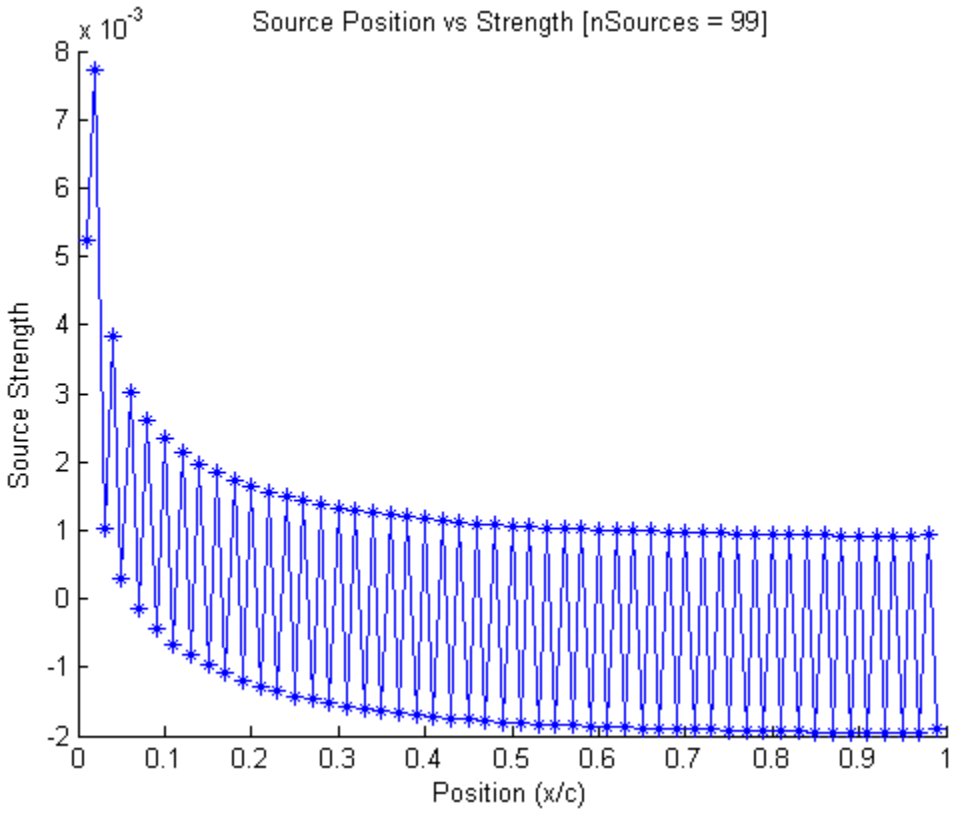
```
end


xyNext(1) = u .* dt + x;
xyNext(2) = v .* dt + y;
end
```

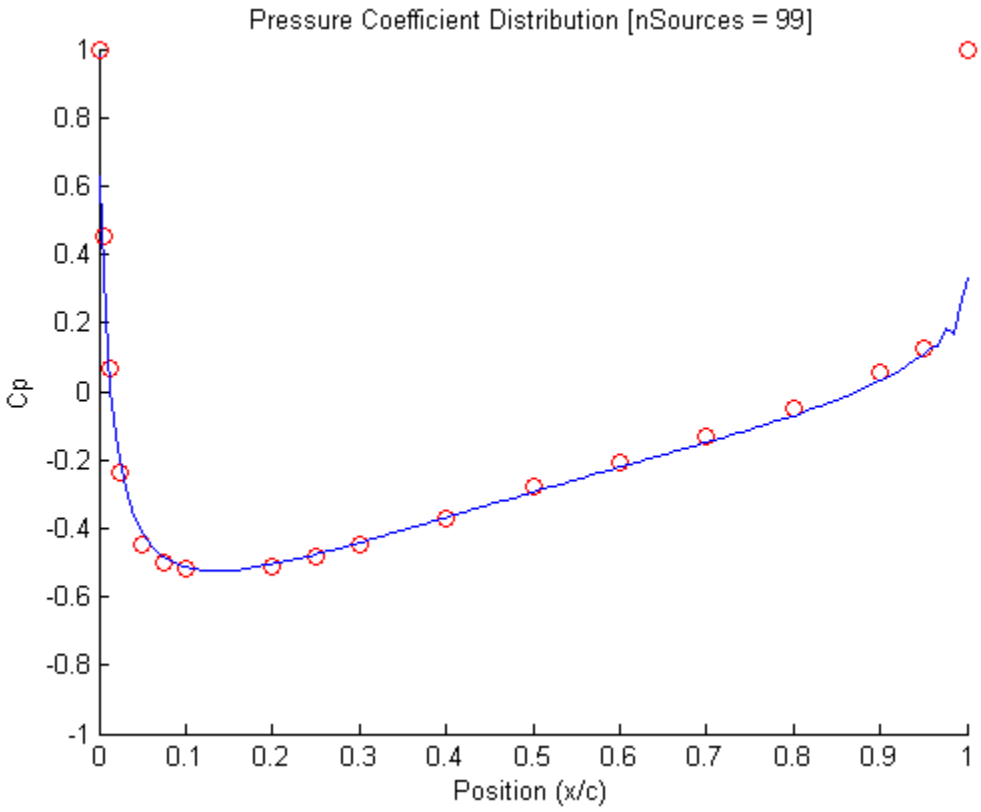
```
%Joel Lubinitsky
%AEE 342 - Project 1a: Analysis of Symmetric Airfoils
%RMS Error
%01/30/15

function errorRMS = p1bErrorRMS(nSinks)
ratioPositionChord = [0 0.005 0.0125 0.025 0.050 0.075 0.10 0.20 0.25 0.30 :
0.1 : 0.90 0.95 1.00]';
coefficientPressureExp = [1.000 0.454 0.067 -0.237 -0.450 -0.498 -0.520 -0.510
-0.484 -0.450 -0.369 -0.279 -0.206 -0.132 -0.049 0.055 0.128 1.000]';
t = 0.15;
xSink = zeros(1, nSinks);
for i = [1 : length(xSink)]
    xSink(i) = i / (nSinks + 1);
end

yAirfoil = (t ./ 0.20) .* (0.2969 .* sqrt(xSink) - 0.1260 .* xSink - 0.3516 .*
xSink .^ 2 + 0.2843 .* xSink .^ 3 - 0.1015 .* xSink .^ 4);

M = zeros(nSinks, nSinks);
for j = [1 : nSinks]
    for i = [1 : nSinks]
        M(j, i) = atan2(yAirfoil(j), (xSink(j) - xSink(i)));
    end
end

R = -yAirfoil';
s = M\R;

T = 10;
dt = 0.01;
N = (T / dt) + 1;
xyAirfoil = zeros(N, 2);
for i = [-0.001, 0.001]
    xyAirfoil(1, :) = [0, i];

    for n = [1 : N - 1]
       xyAirfoil(n + 1, :) = p1bEuler(xyAirfoil(n, :), s, xSink, dt);
    end
end

uAirfoil = 1;
vAirfoil = 0;
for i = [1 : nSinks]
    uAirfoil = uAirfoil + s(i) .* (xyAirfoil(:, 1) - xSink(i)) ./
((xyAirfoil(:, 1)- xSink(i)) .^ 2 + xyAirfoil(:, 2).^ 2);
```

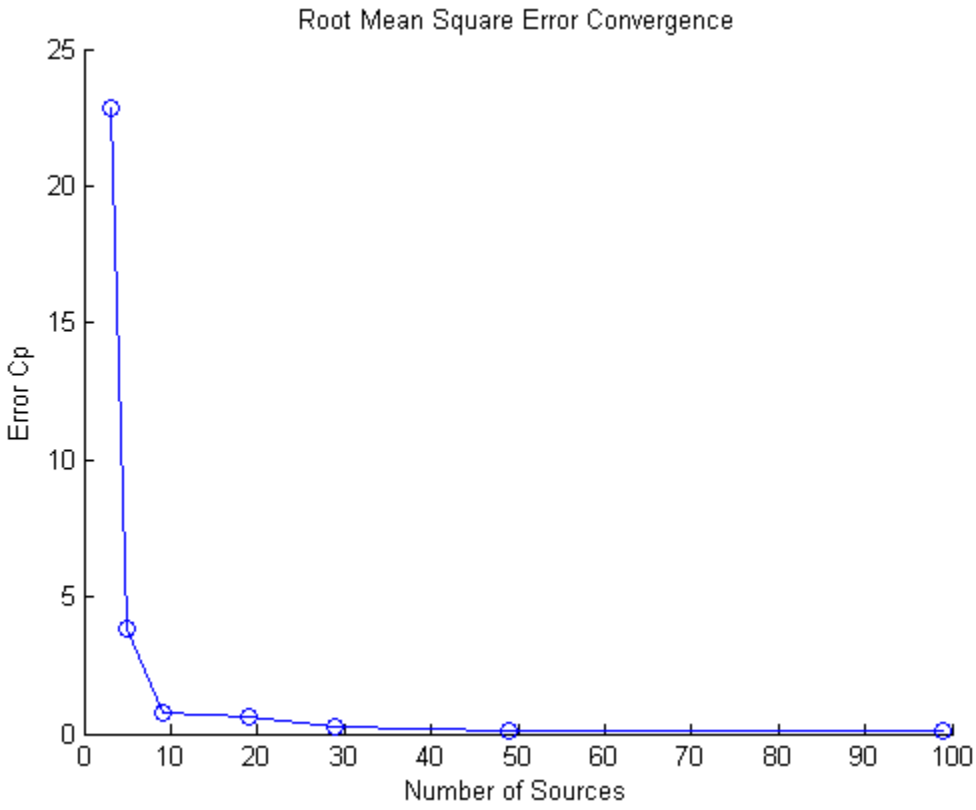
```
    vAirfoil = vAirfoil + s(i) .* xyAirfoil(:, 2)./ ((xyAirfoil(:, 1) -
xSink(i)) .^ 2 + xyAirfoil(:, 2).^ 2);
end

velocityFreestream = 0.9996;
[minEndAirfoil, indexEndAirfoil] = min(abs(xyAirfoil(:, 1) - 1));
qAirfoil = sqrt(uAirfoil .^ 2 + vAirfoil .^ 2);
coefficientPressureSim = 1 - (qAirfoil .^ 2) ./ (velocityFreestream .^ 2);

%Root Mean Square Error
sumDiffSq = 0;
for n = [1 : indexEndAirfoil - 1]
    sumDiffSq = sumDiffSq + (coefficientPressureSim(n)  -
interp1(ratioPositionChord, coefficientPressureExp, abs(xyAirfoil(n, 1)))) .^
2;
end
errorRMS = sqrt(sumDiffSq ./ (indexEndAirfoil - 1));
```

Plots


Field Plot displaying Airfoil, Sources, Streamlines [nSources = 99]
Y
X
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1
1.5
2


Source Position vs Strength [nSources = 99]
x 10^-3
Source Strength
Position (x/c)
8
7
6
5
4
3
2
1
0
-1
-2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Pressure Coefficient Distribution [nSources = 99]
Cp
Position (x/c)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1



References

Matlab Documentation. http://www.mathworks.com/help/matlab/.