Joel Lubinitsky AEE 342
02/06/15 Project 1c

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

Submitted: 02/06/15

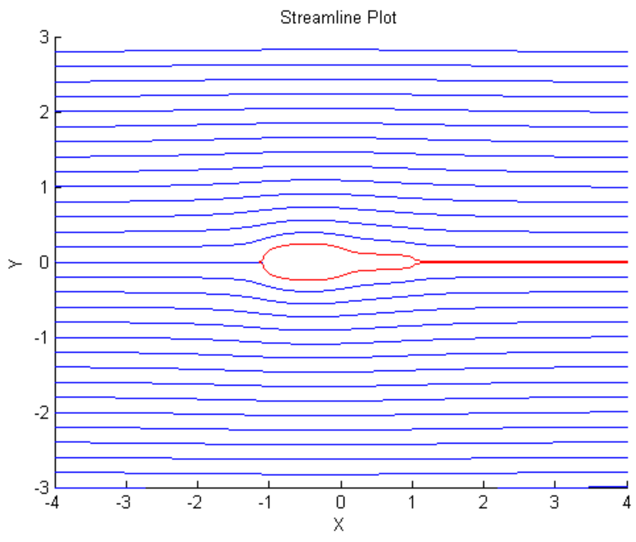
In the simulation of flow behavior, it is important to properly constrain the system with fundamental 'rules' that reflect the laws of the physical system being modelled. These constraints to fluid behavior often seem trivial when observing a physical system, but may require some clever use of mathematical principles in order to force a simulation to behave a certain way. Throughout the course of this investigation of flowfields and their behavior, such principles are applied to model several systems and to fine-tune their properties. The simulated flow was introduced to two different solid bodies. The shapes of these bodies were defined by placing some number of sources and sinks. Sources and sinks are point singularities that repel and attract a simulated flow, respectively. That is, they are points that radially emanate infinitely many vectors with magnitudes equal to the source's or sink's strength, s. This is a valuable construct in the simulation of fluid flows because sources and sinks may be placed in order to induce a certain curvature in the flowfield. As more points are placed, a boundary can become sufficiently defined so as to simulate the normal forces induced on the flowfield by the object being modelled. However, the curvature and force of a physical surface is continuous, while the number of sources/sinks placed must be finite. This issue is addressed in detail throughout this investigation, with visual evaluation of the accuracy of simulated airfoils as well as a comparison with table values of airfoil performance with varying numbers of sources and sinks.

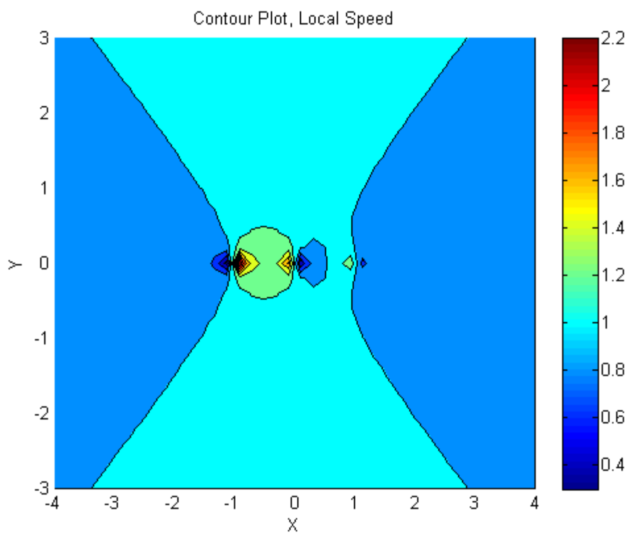
The investigation began with the modelling of an object that appeared similar to a bowling pin. This was a simple model using only three sources/sinks, but provided valuable insight into the nature of flowfields through a number of intuitive visualizations. This flowfield was described by the following equations.

$$u(x,y) = \frac{s_1(x+1)}{(x+1)^2+y^2} + \frac{s_2 x}{x^2+y^2} + \frac{s_3(x-1)}{(x-1)^2+y^2} + 1$$

$$v(x,y) = \frac{s_1 y}{(x+1)^2+y^2} + \frac{s_2 y}{x^2+y^2} + \frac{s_3 y}{(x-1)^2+y^2}$$

where $s_1 = 0.10$, $s_2 = -0.07$, $s_3 = -0.03$. The primary components of these equations are the positions and strengths of the sinks, as well as the velocity of the freestream. The first three terms in each equation are the equations describing the three sinks. The final constant term, which is only found in the equation for u, represents the velocity of the freestream. Here, it can be seen that the flowfield described has a freestream velocity of 1 in the x-direction and has three major changes in curvature. By integrating these velocity functions numerically, the position of the flow on an arbitrary number of streamlines can be determined and plotted. The resulting streamlines and airfoil shape can be seen below. An additional visualization of the simulation that was useful for an intuitive understanding of the system was the contour plot, displaying the velocity distribution across the entire field. These two visualizations of the flow complemented





each other very well, with streamlines describing the movement of the flow near the modelled surface, and the contour plot providing insight into the gradual and abrupt changes in velocity throughout the entire field. On the streamline plot, an additional pair of streamlines is plotted near the center, beginning from the stagnation point at the leading edge of the object. These streamlines trace the surface of the object and reveal much about its shape. This is a strategy that will be employed many times throughout the investigation. On the contour plot, particularly noteworthy areas can be identified by drastic changes in velocity, such as the stagnation point. The general sweeping curves of the plot are informative too, possibly suggesting an apparent pressure gradient throughout field.

Next, the general forms of the velocity field equations were introduced in order to derive equations particular to a NACA 0015 airfoil. Here, several mathematical relations derived from physical characteristics were used to constrain the flowfield. First, the flowfield velocities were defined by

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

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

where n is the total number of sources/sinks and i is the particular source/sink number. Here, the freestream velocity is again set to 1, while remaining velocities will be influenced by the determined source/sink strengths. The first constraint to this system is the equation for the shape of a NACA airfoil, given by

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

where t = 15 for a NACA 0015 airfoil. Although the shape of the airfoil can now be determined, another relation must be applied in order to relate it to the velocity functions. In order to match streamlines to the surface of the airfoil, the system can be constrained by setting the stream function $\psi(x_j, y_j)$ equal to zero at all points on the airfoil. This yields the equation

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

where subscript j refers to a point on the airfoil's surface. Using these relations, the source/sink strengths necessary to model the airfoil can be determined. By choosing enough sources/sinks to plot, the surface can be approximated to a desirable degree of accuracy. However, this method introduced an issue where sources would alternate with sinks in order to simulate a straight line, but ended up resulting in small undulations along the surface of the airfoil. Fortunately, the method used was not the only way to constrain the system. One way to match streamlines to the airfoil shape while avoiding this issue would be to set the velocity component normal to the airfoil surface equal to zero. The physical interpretation of this is intuitive, as it essentially describes the fact that air cannot go through a wing. Mathematically, this would ensure that the slope of the streamline matches that of the airfoil, and can be described by the following relation.

$$[dx, dy] \times [u, v] = 0$$

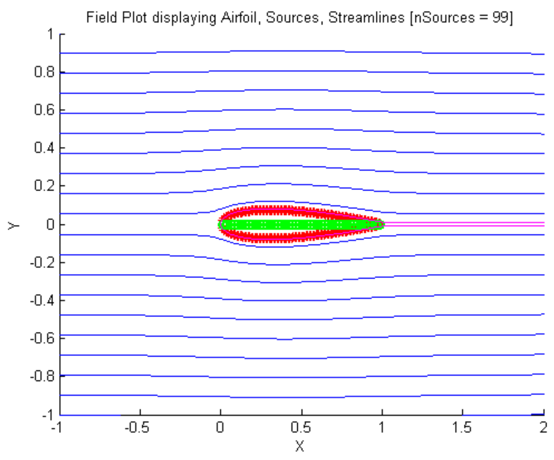
As long as this holds true, the flow velocity cannot intersect the airfoil boundary. Solving this yields

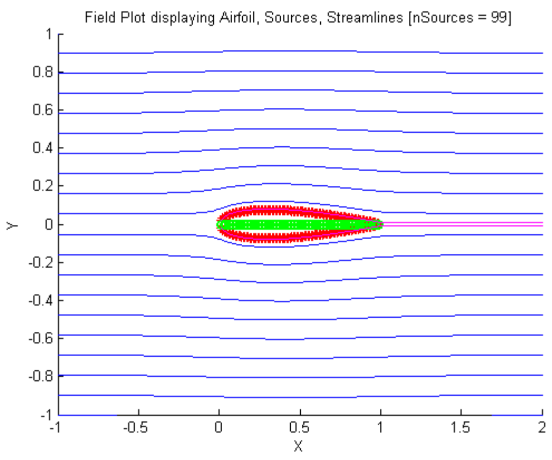
$$\frac{v}{u} = \frac{dy}{dx}$$

Substituting and factoring s_i

$$\left[\frac{y_j}{\left(x_j - x_i\right)^2 + y_j^2} - \frac{x_j - x_i}{\left(x_j - x_i\right)^2 + y_j^2}\frac{dy}{dx}\right] s_i = \frac{dy}{dx}$$

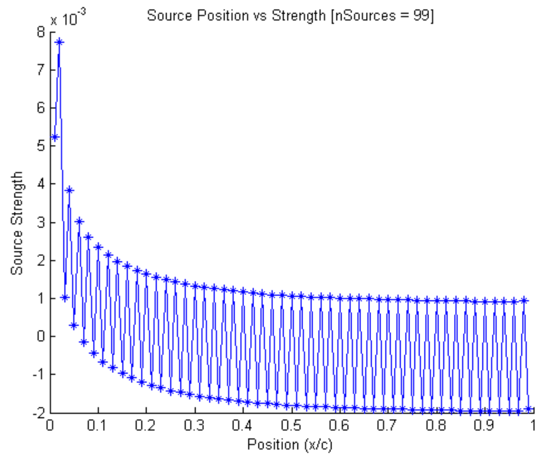
The values of s obtained this way do not exhibit the unwanted behavior of the previous model and appear to simulate flow over the airfoil well, assuming enough points have been plotted. Below is a comparison of the two methods and their outcomes.

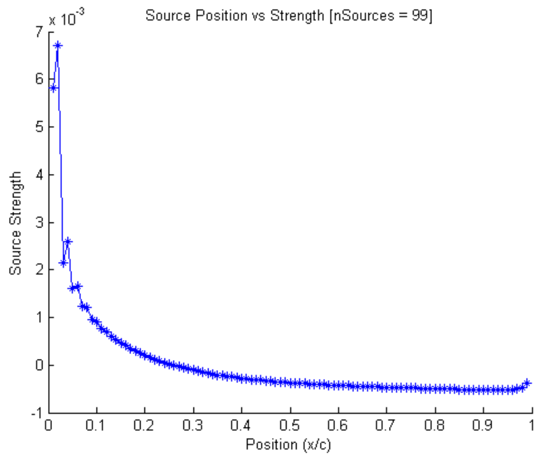


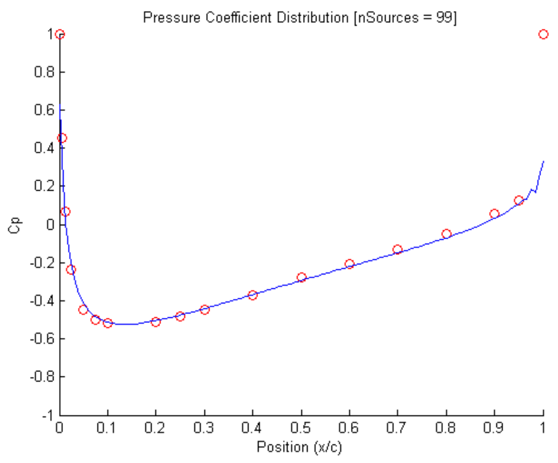


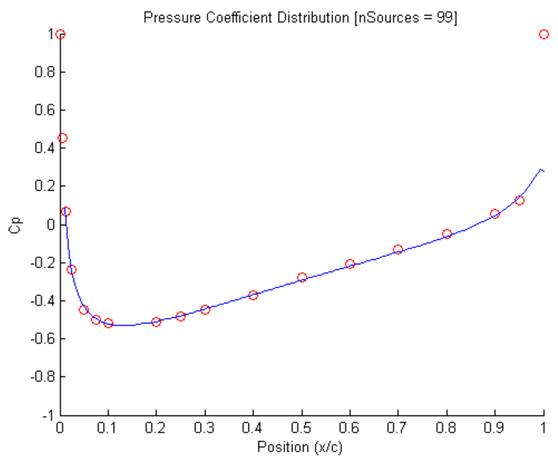
It is immediately clear that these plots are both very similar; almost indistinguishable. In the plot on the left, source/sink strengths were found by setting the stream function equal to zero, and by setting the cross product equal to zero on the right. Clearly, two constraints which are mathematically very different ended up providing a nearly identical outcome. This is because the physical implications of both mathematical principles are very similar, although they are not quite the same. It is this nuance of the two methods that will be discussed next.

The two plots below correspond to the two plots above, respectively. Here it is quite clear that while the streamlines appear similar, they are in fact behaving very differently. In the original model on the left, relatively straight sections of the airfoil are modelled by many alternating sources and sinks, averaging out to approximately the desired value. Conversely, the newer model on the right is forced to obey the condition that streamlines cannot penetrate the surface of the airfoil, and thus cannot undulate. This is reflected directly in the plot below, showing a smooth distribution of sources and sinks along the length of the airfoil. Airfoils in real life are in fact smooth since their manufacturers try to avoid having small bumps and undulations. For this reason, the second of the two methods is a better representation of physical properties and is thus a better simulation.





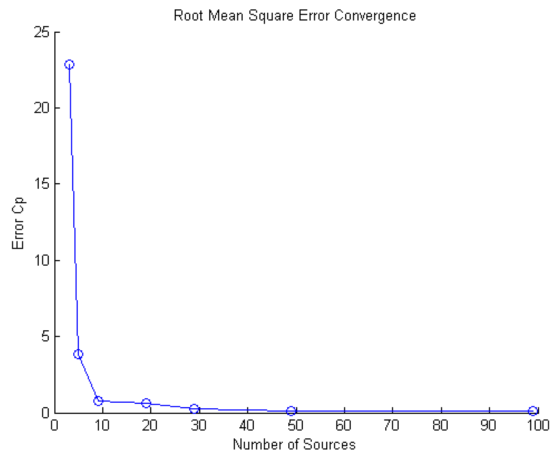


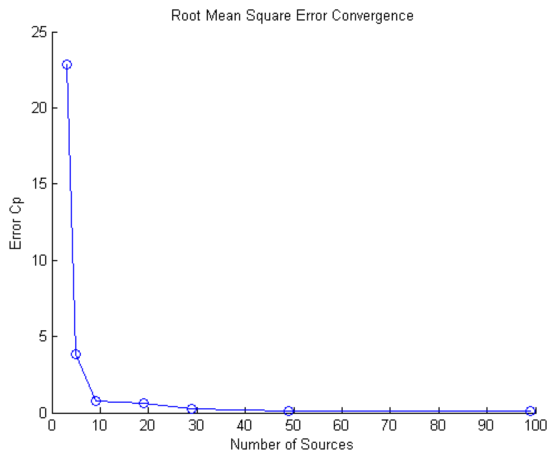


In order to evaluate the accuracy of the model, pressure coefficients were calculated along the surface of each airfoil and then compared to interpolated table values. A superficial comparison can be seen above, where calculated coefficient of pressure is shown by the curve, and table values are shown as circles on the plot. The pressure coefficient was calculated 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. It is evident from both plots that the simulation with 99 sources/sinks is quite accurate, and that the difference between the two constraining methods is not substantial. One difference to be pointed out, however, is that undulations exist at the end of the left plot, but do not on the right one. This is likely because of the undulations in the surface of the airfoil. A second difference, which is unrelated to the methods used, is that the first several values are cut off on the second plot. This is only because of an adjustment made to the point of emanation for the airfoil streamline. This was done only for cosmetic purposes of the streamline plot and is unrelated to the current analysis.





The final two plots above were created in order to perform a more exhaustive analysis of the different airfoil models depending on the number of sources/sinks used. The values plotted here are the root mean square error of the computed coefficients of pressure as compared to table values, given by

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

Since the plots showing the coefficients of pressure were almost identical, it is expected that the computed errors would be very similar. This does appear to be the case and both appear to converge to the accepted solution as the number of sources/sinks is increased. However, returns begin to diminish quite rapidly as sources/sinks are added, so a reasonable degree of precision should be decided upon.

In conclusion, this investigation has evaluated the efficacy of modelling flowfields by using sources and sinks to simulate interactions with solid bodies. This methodology has proven itself effective in the case that sufficient sources/sinks have been plotted such that the body may begin to appear continuous instead of as a set of discreet points. However, it has also been shown that the quality of the simulation is influenced by other factors such as the theoretical constraints governing its behavior. The specific mathematical language used to communicate physical principles to the program is critical to the proper behavior of the flowfield and to fine-tuning its results.

Calculations

```
%Joel Lubinitsky
%AEE 342 - Project 1c: Analysis of Symmetric Airfoils
%02/06/15

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);
dydxAirfoil = (t ./ 0.20) .* ((0.14845 .* xSink .^ -0.5) - (0.1260) - (0.7032
.* xSink) + (0.8529 .* xSink .^ 2) - (0.4060 .* xSink .^ 3));

M = zeros(nSinks, nSinks);
for j = [1 : nSinks]
    for i = [1 : nSinks]
        M(j, i) = (yAirfoil(j) / ((xSink(j) - xSink(i)) ^ 2 + yAirfoil(j) ^
2)) - (((xSink(j) - xSink(i)) / ((xSink(j) - xSink(i)) ^ 2 + yAirfoil(j) ^
2)) * ((t / 0.20) * ((0.14845 * xSink(j) ^ -0.5) - (0.1260) - (0.7032 *
xSink(j)) + (0.8529 * xSink(j) ^ 2) - (0.4060 * xSink(j) ^ 3))));
    end
end

R = dydxAirfoil';
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 = [-yAirfoil(1), yAirfoil(1)]
    xyAirfoil(1, :) = [xSink(1), 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);
```

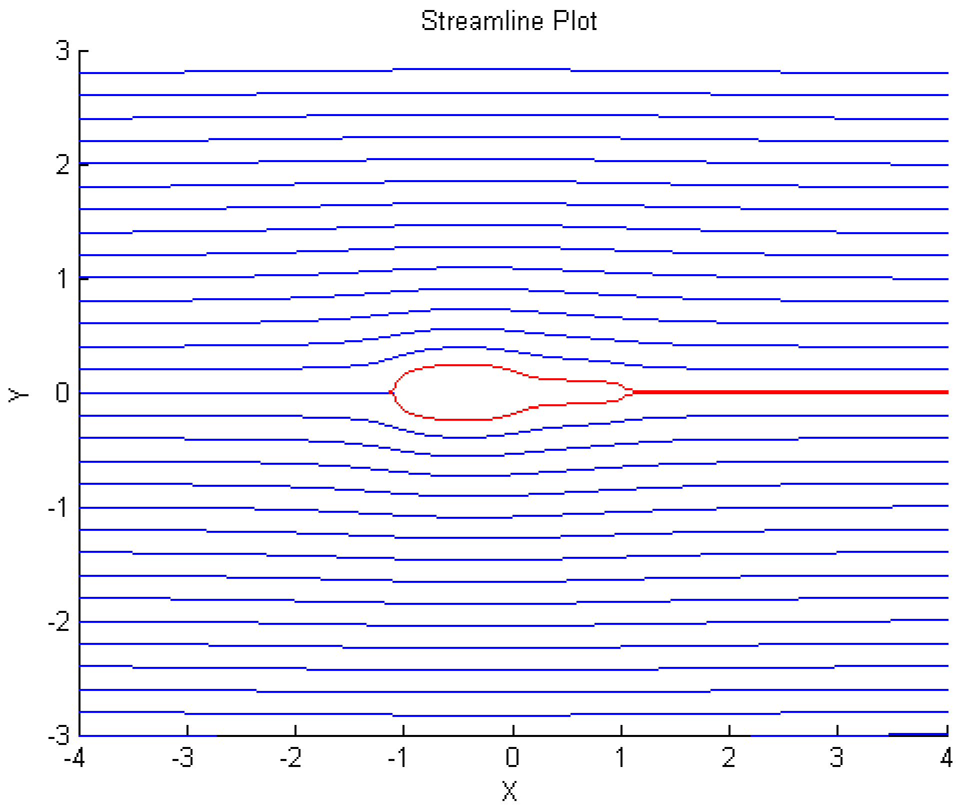
```
u = 1;
v = 0;
for i = [1 : length(s)]
    u = u + s(i) .* (x - xSink(i)) ./ ((x - xSink(i)) .^ 2 + y .^ 2);
    v = v + s(i) .* y ./ ((x - xSink(i)) .^ 2 + y .^ 2);
end


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));
```

Joel Lubinitsky
02/06/15

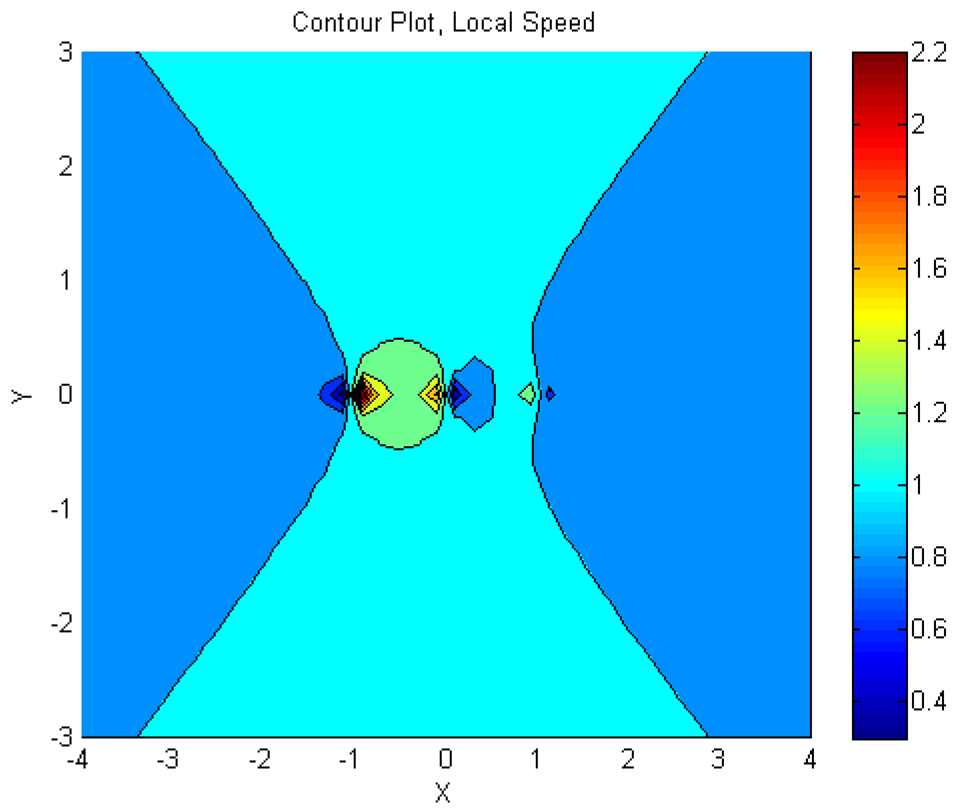
AEE 342
Project 1c

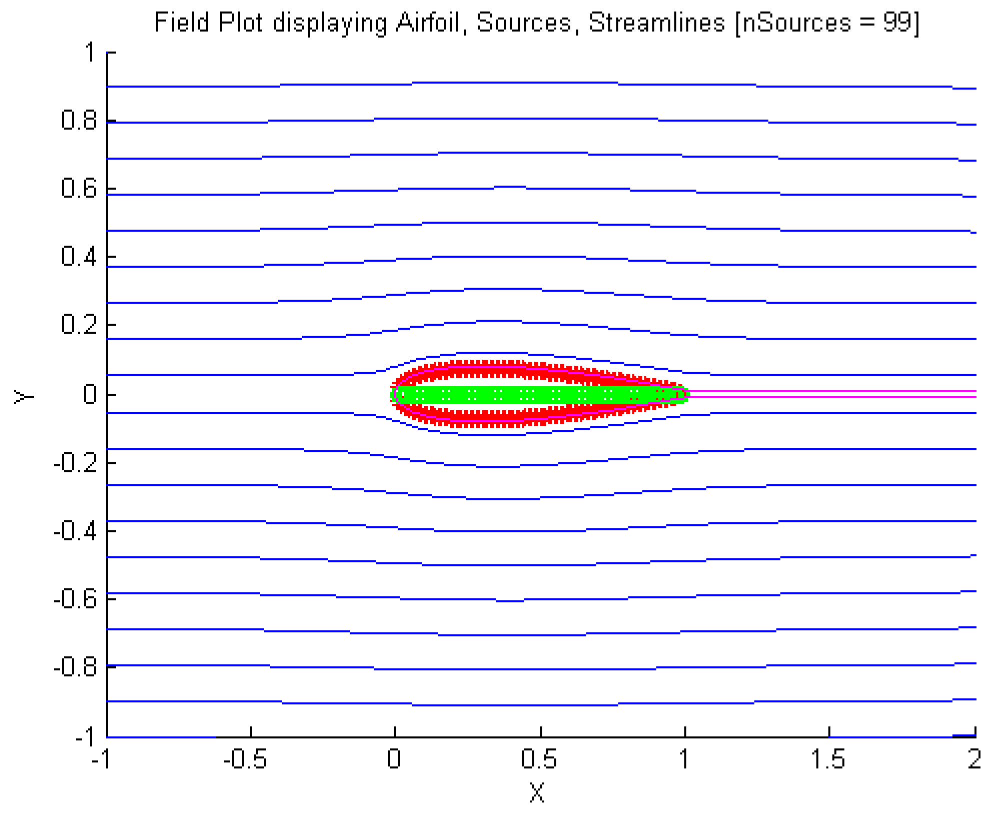
Plots

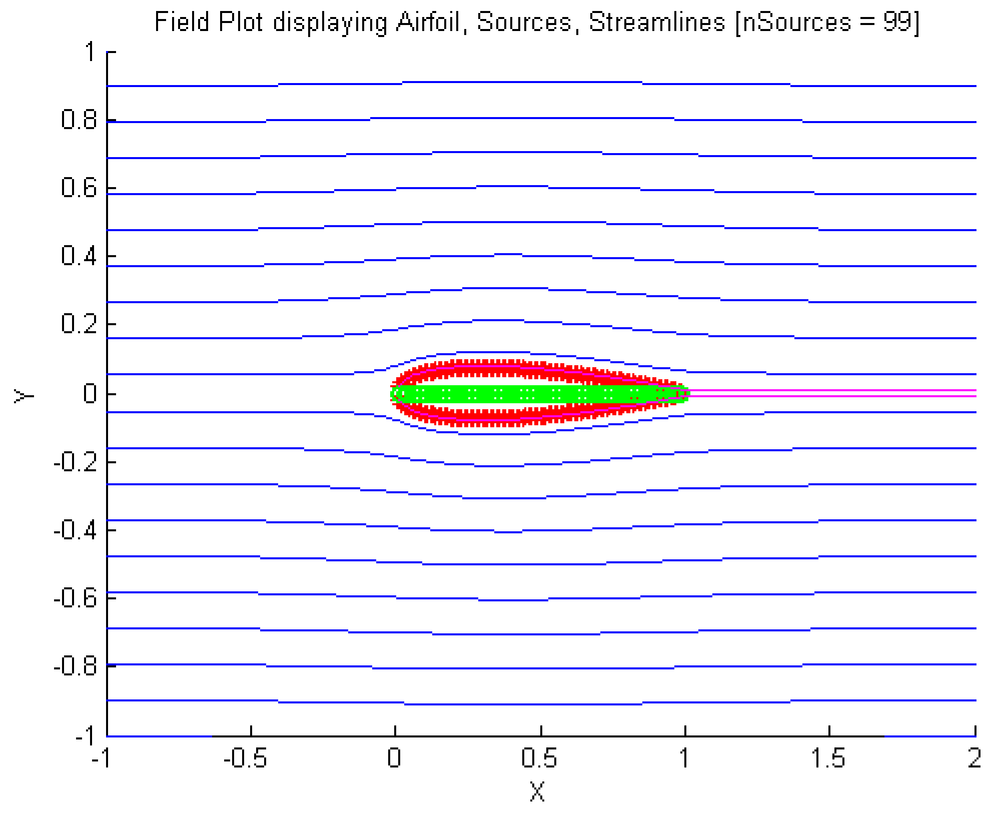


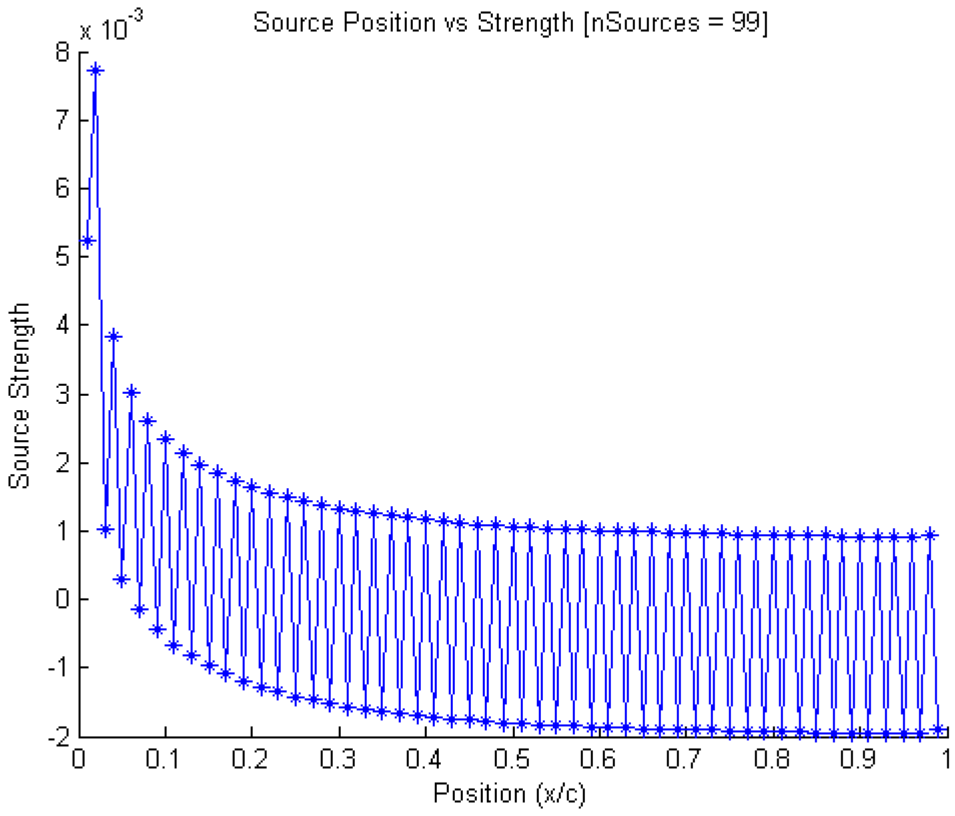
Joel Lubinitsky
02/06/15

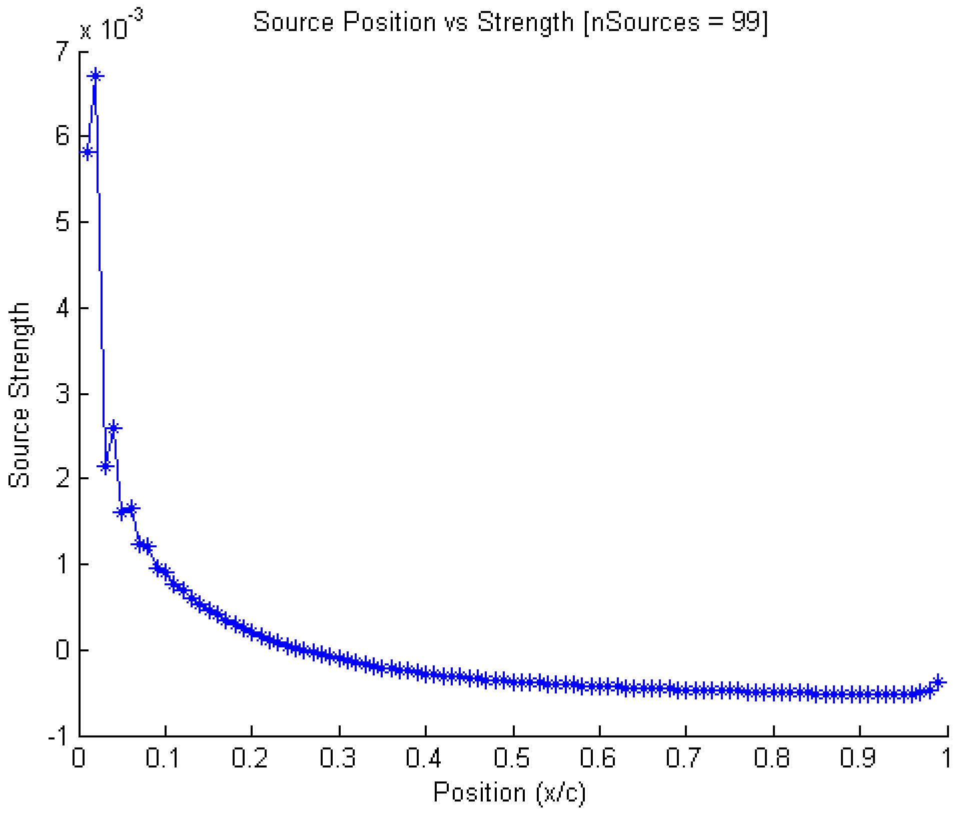
AEE 342
Project 1c

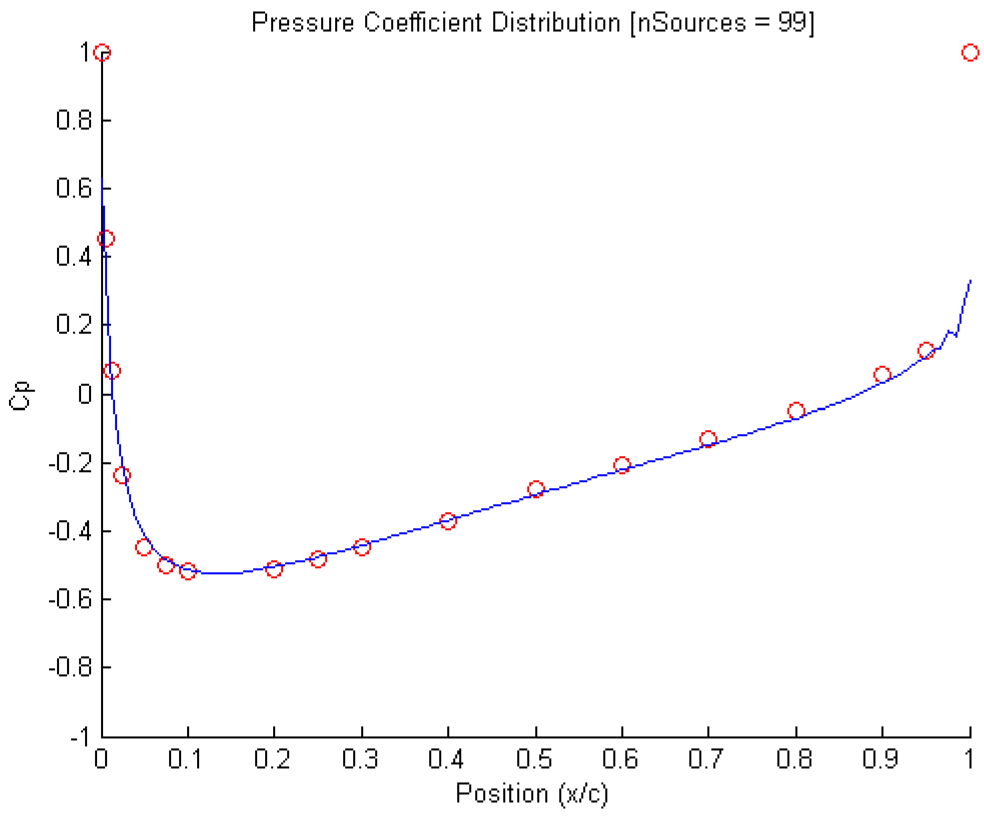


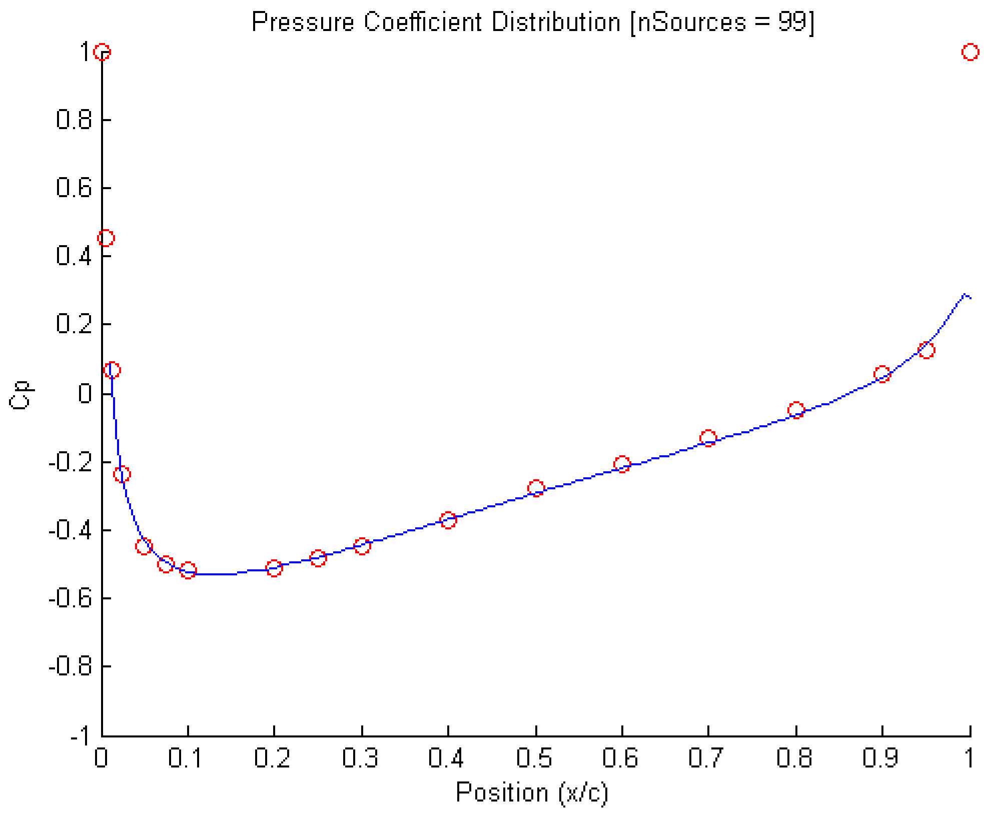

Field Plot displaying Airfoil, Sources, Streamlines [nSources = 99]
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
Y
-1
-0.5
0
0.5
1
1.5
2
X


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


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

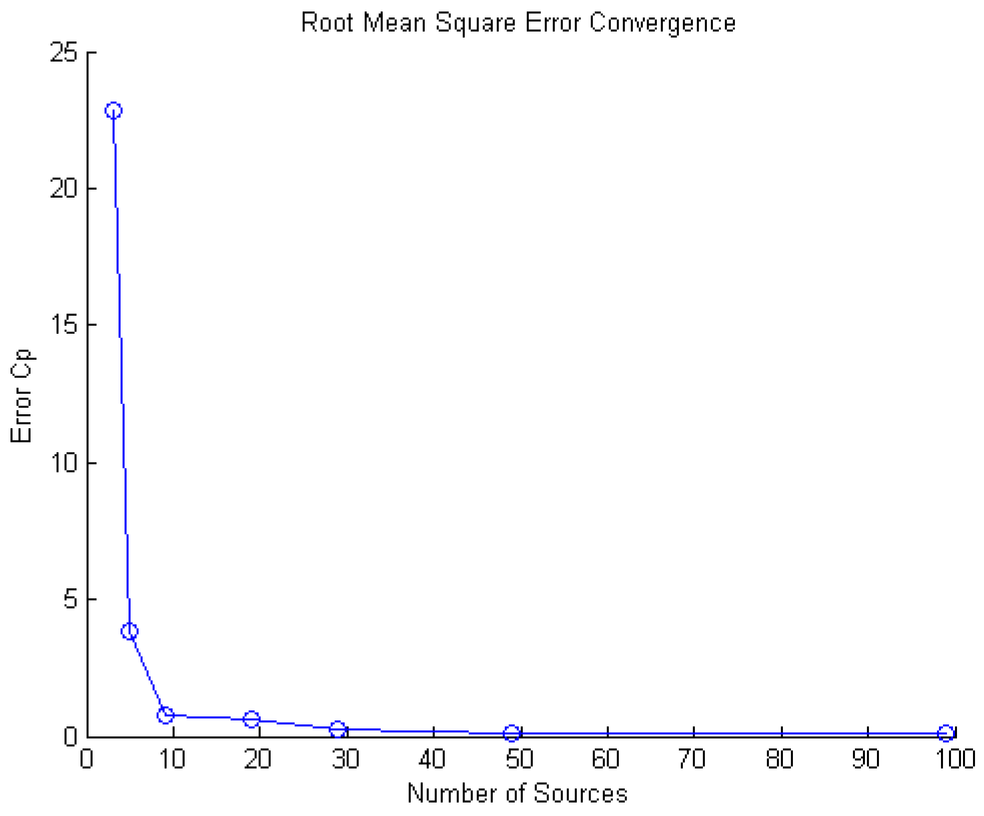

Source Position vs Strength [nSources = 99]
x 10^-3
Source Strength
Position (x/c)
7
6
5
4
3
2
1
0
-1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

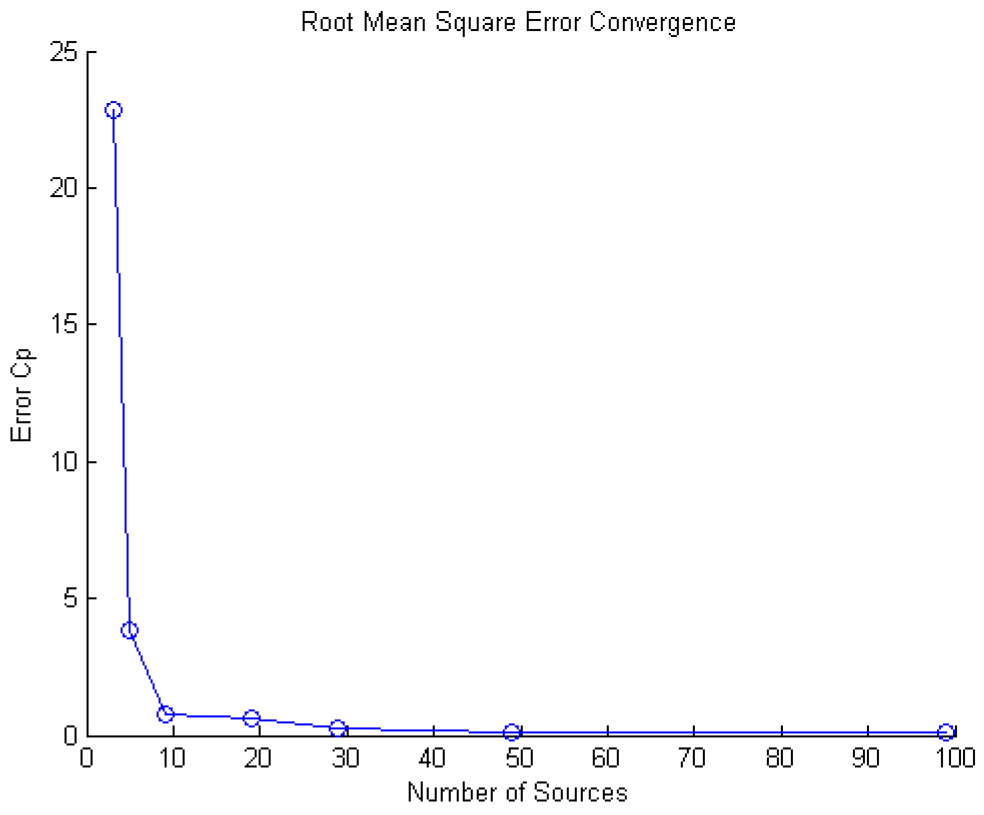

Pressure Coefficient Distribution [nSources = 99]
Cp
Position (x/c)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


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

Joel Lubinitsky
02/06/15

AEE 342
Project 1c





References

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