An Analysis of the Pressure Distribution Over an Ellipse Using the Source Panel Method

Matthieu Lu

A = 4.2, C = 0.8

Table of Contents

Overview

In this project, the source panel method will be used in order to estimate the pressure distribution across an arbitrary symmetric elliptical airfoil.

The source panel method is a method for designers to start with a shape and then solve for the distribution of singularities which, when combined with a uniform stream, produce a flow over the given body. This allows them to take advantage of modern technology's computing power and not restrict designs to elementary flow conditions.

The source panel begins by combining a lot of source lines with infinitesimally small strengths into a large source sheet. These sheets are then arranged into the general shape of an airfoil, and when analyzed from the side, are seen as "panels".

When analyzing these panels, one can analyze and sum up the potential contributions from each panel, also known as jth panels has to a singular panel, called the ith panel. At the ith panel's control point, the normal component of the flow velocity must equal 0. Using this boundary condition, we can sum up the velocity contributions from all of the other panels by using a function with respect to their potentials, the component of velocity from the free stream normal to the ith panel, and the velocity contribution normal to the ith panel from the ith panel itself. When performing this analysis by treating each panel as the ith panel, a linear system can be generated that will help solve for the actual source strength of each panel. With these strengths, the tangential velocity over each panel can be calculated, which in turn allows for the total surface velocity over each control point to be calculated. This can then be related back to a coefficient of pressure, and if these are plotted over the entire airfoil, one can determine the pressure distribution over the entire airfoil.

Governing Equations and Conditions

Analytical Method:

Ellipse

- $$\frac{\mathrm{Xb}^2}{\left(a+\frac{c^2}{a}\right)^2}+\frac{\mathrm{Yb}^2}{\left(a-\frac{c^2}{a}\right)^2}=1$$

Streamline Function

- $\mu = 4\,x^2\,y^2 + \left(4\,c^2 - x^2 + y^2\right)^2$
- $\gamma = -\mathrm{atan}\left(\frac{2\,x\,y}{4\,c^2 - x^2 + y^2}\right)$
- $\psi = \frac{U\,\left(y\,\left(a^2+c^2\right) - \mu^{1/4}\,\sin\left(\frac{\gamma}{2}\right)\,\left(a^2-c^2\right)\right)}{2\,c^2}$, plug in mu and gamma from above.

Velocity Components

- $u = \frac{U\,\left(\mu^{1/4}\,\left(a^2+c^2\right) - \left(a^2-c^2\right)\,\left(y\,\sin\left(\frac{\gamma}{2}\right) + \cos\left(\frac{\gamma}{2}\right)\,|x|\right)\right)}{2\,c^2\,\mu^{1/4}}$
- $v = \frac{U\,\left(a^2-c^2\right)\,\left(y\,\cos\left(\frac{\gamma}{2}\right) - \sin\left(\frac{\gamma}{2}\right)\,|x|\right)}{2\,c^2\,\mu^{1/4}}$
    - Plug in mu and gamma from above.

Analytical Coefficient of Pressure

- $\mathrm{Cp} = 1 - \frac{u^2+v^2}{U^2}$, plot a distribution over the x length of the airfoil.

Source Panel Method:

Ellipse

- $\frac{\mathrm{Xb}^2}{\left(a+\frac{c^2}{a}\right)^2}+\frac{\mathrm{Yb}^2}{\left(a-\frac{c^2}{a}\right)^2}=1$, convert to polar coordinates to find the angles of panel vertices, and use those vertices to find panel control points, panel angle, and panel normal angle.

Integral

- $I_{\mathrm{ij}}=\frac{\ln\left(\frac{\mathrm{Sj}^2+2\,A\,\mathrm{Sj}+B}{B}\right)\,(D-A\,C)}{2\,E}-C\,\left(\operatorname{atan}\left(\frac{A+\mathrm{Sj}}{E}\right)-\operatorname{atan}\left(\frac{A}{E}\right)\right)$

Integral's Components

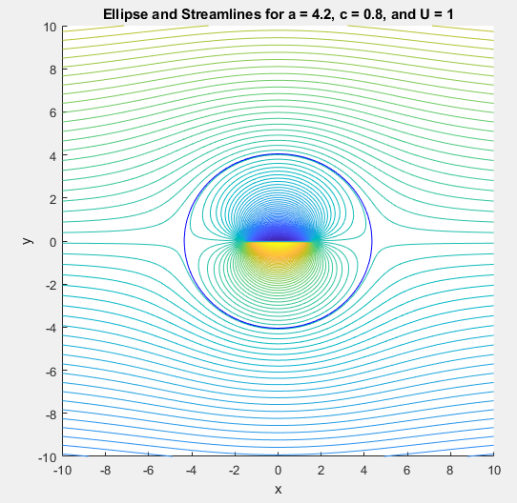
- $A=\cos\left(\phi_j\right)\,\left(X_j-x_i\right)+\sin\left(\phi_j\right)\,\left(Y_j-y_i\right)$
- $B=\left(X_j-x_i\right)^2+\left(Y_j-y_i\right)^2$
- $C=\sin\left(\phi_i-\phi_j\right)$
- $D=\sin\left(\phi_i\right)\,\left(X_j-x_i\right)-\cos\left(\phi_i\right)\,\left(Y_j-y_i\right)$
- $E=\sqrt{B-A^2}$
- $\mathrm{Sj}=\sqrt{\left(X_j-X_{\mathrm{j1}}\right)^2+\left(Y_j-Y_{\mathrm{j1}}\right)^2}$

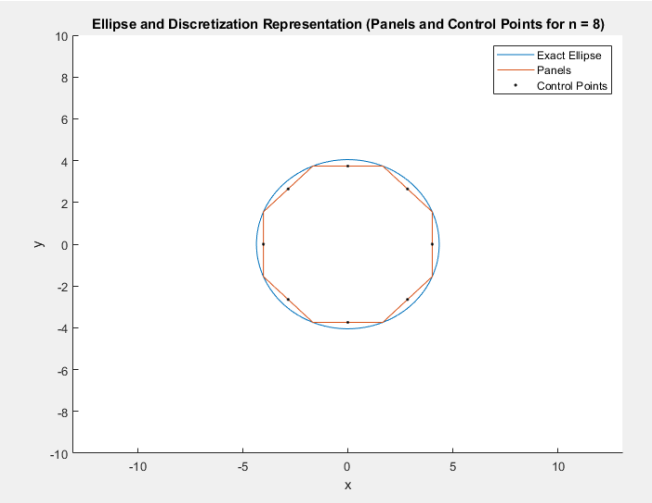
Surface Velocity at ith Control Point from Free Stream and jth Panels

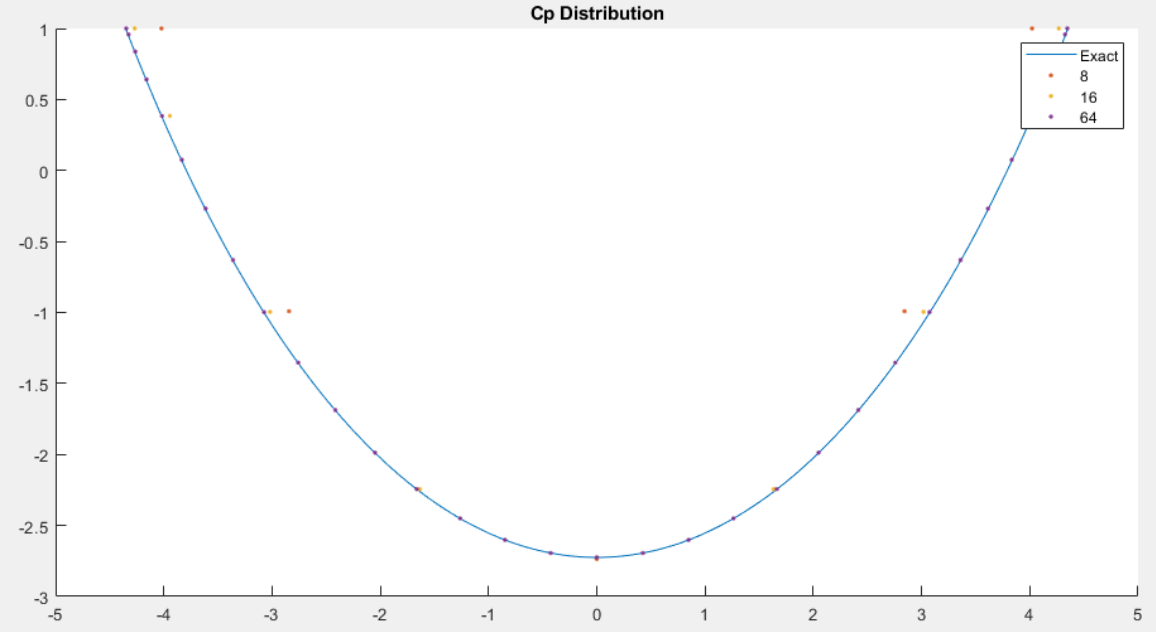
- $\frac{\lambda_i}{2}+\sum_{\substack{j=1\\(j\neq 1)}}^{n}\frac{\lambda_j}{2\pi}I_{i,j}+V_\infty\cos\beta_i=0$

- $\sum I_{ij}\,\lambda_j = -2\,\pi\,V_\infty\,\cos(\beta_i)$, solve for λ_j using matrix methods
- $$V_i = V_{\infty,s} + V_s = V_\infty \sin\beta_i + \sum_{j=1}^{n} \frac{\lambda_j}{2\pi} \int_j \frac{\partial}{\partial s}(\ln r_{ij})\, ds_j$$
  - $$\int_j \frac{\partial}{\partial s}(\ln r_{ij}) ds_j = \frac{D - AC}{2E} \ln \frac{S_j^2 + 2AS_j + B}{B} - C\left(\tan^{-1}\frac{S_j + A}{E} - \tan^{-1}\frac{A}{E}\right)$$
- $$C_{p,i} = 1 - \left(\frac{V_i}{V_\infty}\right)^2$$

Results


Ellipse and Streamlines for a = 4.2, c = 0.8, and U = 1
x
y


Ellipse and Discretization Representation (Panels and Control Points for n = 8)
Exact Ellipse
Panels
Control Points
x
y


Cp Distribution
Exact
8
16
64

Conclusion

The source panel method, with increasing panels, would more accurately represent the analytical Coefficient of Pressure calculations. This is most likely due to the fact that with more panels, it more accurately models the actual shape of the airfoil, which in turn would also more accurately represent the surface velocity over each panel, leading to a more similar Cp graph.

When comparing the code between the analytical and source panel solutions, the code to the analytical solution is much simpler than the source panel. Once given the streamline function, and from that the u and v components of velocity are derived, it's a simple generation of an array from these equations to solve for Cp.

However, for the source panel, a large amount of the code is determining the panel and control point geometries, as well as the angles associated with each panel. However, once those are all figured out, it's just a matter of looping through every ith panel and jth panel to generate an I_ij matrix using the integral component equations which are given, which can then be used to eventually solve for the different source strengths for each panel, and those can then be plugged directly into an equation for velocity and Cp.

It must be noted that the analytical code seems simpler, but the functions that define the streamlines, u, and v are actually quite complex, and not very flexible if the shape changes. Deriving those equations again would be very difficult for a non-elliptical shape. Theoretically, the code for the source panel ought to be applicable to any shape, as long as the shape can be defined with a function.

If this experiment was conducted in the real world, you'd likely have a pretty accurate Cp plot for the first half of the ellipse, but once the ellipse begins to recede and the boundary layer

begins to separate/become turbulent, the Cp graph should look very different. Realistically, the pressure coefficient shouldn’t dip so far below 0.

Appendix 1: Code

```
clear all
clc
close all

U = 1;
a = 4.2;
c = .8;

xst = (a+((c^2)/a));
yst = (a-((c^2)/a));

ellipse = @(x,y) (x./xst).^2 + (y./yst).^2 - 1;

%Streamline

[X1,Y1] = meshgrid(-10:.1:-.01,-10:.1:10);
gamma1 = -atan2((2*X1.*Y1) , ((X1.^2)-(Y1.^2)-(4*(c^2))));
mu1 = (4*(X1.^2).*(Y1.^2)) + ((4*(c^2))-(X1.^2)+(Y1.^2)).^2;
psi1 = (U/(2*(c^2))) * ((((a^2)+(c^2))*Y1) - (((a^2)-(c^2))*(mu1.^(1/4)).*sin(gamma1./2)));

[X2,Y2] = meshgrid(0:.1:10,-10:.1:10);
gamma2 = atan2((2*X2.*Y2) , ((X2.^2)-(Y2.^2)-(4*(c^2))));
mu2 = (4*(X2.^2).*(Y2.^2)) + ((4*(c^2))-(X2.^2)+(Y2.^2)).^2;
psi2 = (U/(2*(c^2)))*((((a^2)+(c^2))*Y2)-(((a^2)-(c^2))*(mu2.^(1/4)).*sin(gamma2./2)));

[X,Y] = meshgrid(-10:.1:10,-10:.1:10);
psi = [psi1 psi2];
gamma = [gamma1 gamma2];
mu = [mu1 mu2];

%Coefficient of Pressure

%Body

xtop1 = linspace(-xst,-.01,xst*100);
xtop2 = linspace(0,xst,(xst*100)+1);
xtop  = linspace(-xst,xst,2*(xst*100)+1);
ytop1 = sqrt(1-(xtop1/xst).^2)*yst;
ytop2 = sqrt(1-(xtop2/xst).^2)*yst;
ytop  = sqrt(1-(xtop/xst).^2)*yst;

gammatop = [-atan2(2*xtop1.*ytop1,((xtop1.^2)-(ytop1.^2)-4*c^2))
atan2(2*xtop2.*ytop2,((xtop2.^2)-(ytop2.^2)-4*c^2))];
```

```
mutop = (4*(xtop.^2).*(ytop.^2))+(4*c^2-xtop.^2+ytop.^2).^2;

u =
(U./(2*c^2*mutop.^(1/4))).*((a^2+c^2)*mutop.^(1/4)-(a^2-c^2)*(abs(xtop).*cos(gammatop/2)+
ytop.*sin(gammatop/2)));
v = U.*(a^2-c^2)*(ytop.*cos(gammatop/2)-abs(xtop).*sin(gammatop/2))./(2*c^2*mutop.^(1/4));
Cp = 1 - (u.^2 + v.^2)/(U^2);

subplot(2,2,3)
hold on
plot(xtop,Cp)

x = [8 16 64]
for y = [1:3]

   n = x(y)

   for i = [1:n]
      %ith Panel Vertex Angles (ccw)
      angle_i = (360/(2*n)) - (360/n)*(i-1);
      angle_i_1 = (360/(2*n)) - (360/n)*(i);

      %ith Panel Vertex Coordinates
      X_i(i) = xst * cosd(angle_i);
      X_i(i+1) = xst * cosd(angle_i_1);
      Y_i(i) = yst * sind(angle_i);
      Y_i(i+1) = yst * sind(angle_i_1);

      %ith Panel Midpoint
      x_i(i) = (X_i(i+1) + X_i(i))/2;
      y_i(i) = (Y_i(i+1) + Y_i(i))/2;

      %ith Panel Angle(phi) and Normal Angle(beta)
      rise_i = Y_i(i+1) - Y_i(i);
      run_i = X_i(i+1) - X_i(i);
      phi_i(i) = wrapTo360(atan2d(rise_i,run_i));
      beta_i(i) = wrapTo360(phi_i(i) + 90);

      %Placeholder
      p(i,1) =  -U*2*pi*cosd(beta_i(i));

      for j = 1:n
         %jth Panel Vertex Angles (ccw)
         angle_j = (360/(2*n)) - (360/n)*(j-1);
         angle_j_1 = (360/(2*n)) - (360/n)*(j);
```

```
        %jth Panel Vertex Coordinates
        X_j(j) = xst * cosd(angle_j);
        X_j(j+1) = xst * cosd(angle_j_1);
        Y_j(j) = yst * sind(angle_j);
        Y_j(j+1) = yst * sind(angle_j_1);

        %jth Panel Angle(phi) and Normal Angle(beta)
        rise_j = Y_j(j+1) - Y_j(j);
        run_j = X_j(j+1) - X_j(j);
        phi_j(j) = wrapTo360(atan2d(rise_j,run_j));
        beta_j(j) = wrapTo360(phi_j(i) + 90);

        %Integration Constants
        A(i,j)  = -(x_i(i) - X_j(j))*cosd(phi_j(j)) - (y_i(i) - Y_j(j))*sind(phi_j(j));
        B(i,j)  = (x_i(i)-X_j(j))^2 + (y_i(i)-Y_j(j))^2;
        C(i,j)  = sind(phi_i(i) - phi_j(j));
        D(i,j)  = (y_i(i)-Y_j(j))*cosd(phi_i(i)) - (x_i(i)-X_j(j))*sind(phi_i(i));
        Sj(i,j) = sqrt((X_j(j+1)-X_j(j))^2 + (Y_j(j+1)-Y_j(j))^2);
        E(i,j)  = sqrt(B(i,j) - A(i,j)^2);

        I(i,j)  = (C(i,j)/2)*log((Sj(i,j)^2+(2*A(i,j)*Sj(i,j))+B(i,j))/B(i,j)) +
((D(i,j)-(A(i,j)*C(i,j)))/E(i,j))*(atan((Sj(i,j)+A(i,j))/E(i,j))-atan(A(i,j)/E(i,j)));
        if i == j
            I(i,j) = pi;
        end
      end
    end

    %Lambda
    lambda = linsolve(I,p);
    lambdaRatio = lambda./(2*pi*U);



    for i = 1:n
      V(i) = U*sind(beta_i(i));
      for j = 1:n
        if i ~= j
          V(i) = V(i) +
(lambda(j)/(2*pi))*((((D(i,j)-A(i,j)*C(i,j))/(2*E(i,j)))*log((Sj(i,j)^2+(2*A(i,j)*Sj(i,j))+B(i,j))/B(i,
j))) - C(i,j)*(atan((Sj(i,j)+A(i,j))/E(i,j)) - atan(A(i,j)/E(i,j))));
        end
      end
    end

    Cp_i = 1 - (V/U).^2;
```

```
    subplot(2,2,3)
    hold on
    plot(x_i,Cp_i,'.')
    title('Cp Distribution')


end

legend('Exact','8','16','64')
hold off

subplot(2,2,1)
hold on
axis equal
fimplicit(ellipse, [-10 10 -10 10],'b');
contour(X,Y,psi,[100])
title('Ellipse and Streamlines for a = 4.2, c = 0.8, and U = 1')
xlabel('x')
ylabel('y')
hold off

subplot(2,2,2)
hold on
axis equal
fimplicit(ellipse, [-10 10 -10 10]);
plot(X_i,Y_i)
plot(x_i,y_i,'k.','MarkerSize',5)
title('Ellipse and Discretization Representation (Panels and Control Points for n = 64)')
xlabel('x')
ylabel('y')
legend('Exact Ellipse','Panels','Control Points')
hold off
```