FLUID FLOW AROUND AN AIRFOIL

by

Jim Finnegan

Final Project

ES-55

Professor Chapra

December 22, 2017

Tufts University

Medford, MA

TABLE OF CONTENTS

DESCRIPTION

This program models the flow of a fluid with a constant horizontal velocity around a symmetrical airfoil. The script takes a streamline matrix defined in Microsoft Excel and calculates the horizontal velocity, vertical velocity, and total velocity of the fluid using discrete and regression methods. It also uses plots to display streamlines visually.

INPUTS

There is only one input for the program:

- An Excel file containing a matrix whose entries are the values of streamlines around an object

OUTPUTS

The program outputs the following:

- Contour plot of the streamline data
- Horizontal, vertical, and total fluid velocity plots
- Surface regression for streamline data
- Horizontal, vertical and total fluid velocity plots derived from the surface regression
- Quiver plot of streamline data
- Streamline plot with animated particles moving with the fluid flow

NUMERICAL METHODS

The following numerical methods were used in this program:

- Numerical solution to Laplace's equation in Excel
- User interface – choice between analyzing airfoil or circle
- Numerical differentiation (gradient)
- Polynomial Surface Regression (cftool)
    - Goodness of fit statistics: Root mean squared error (standard error), coefficient of determination
- Solving first order partial differential equations generated in the surface regression (differentiate)
- Animation (streamparticles)

VALIDATION OF RESULTS

This program is a good representation of idealized fluid flow around a smooth object. Objects with sharp corners or with a rough surface could not be represented by this model. Some corners are noticeable in the objects, but that error could be minimized by using a larger matrix.

The program only represents flow that has no turbidity, viscosity, or separation, an idealized case. Although the program represents an idealized case, it is replicated under certain conditions. Reynold's number is a dimensionless parameter that is often used to describe the behavior of a flow Flows with Reynold's numbers less than approximately 2100 are said to be laminar, meaning that the flow does not have any turbulence. Flow separation occurs at different Reynold's numbers for different objects, but a general case is shown in Figure 1.

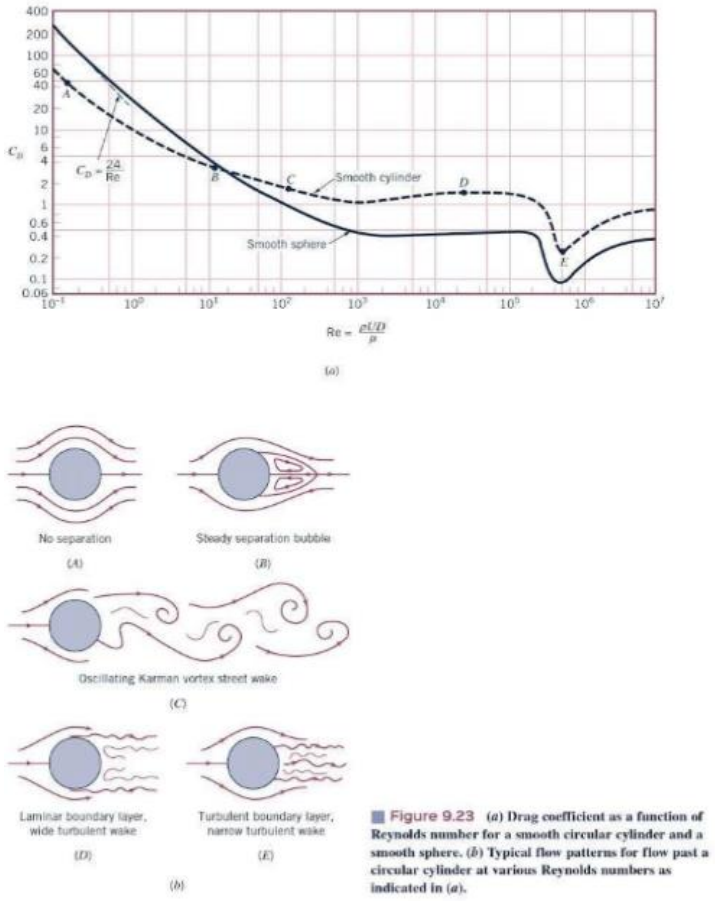


Figure 9.23 (a) Drag coefficient as a function of Reynolds number for a smooth circular cylinder and a smooth sphere. (b) Typical flow patterns for flow past a circular cylinder at various Reynolds numbers as indicated in (a).

Figure 1: Reynold's Number and Flow Separation (Munson et al. p. 524)

The maximum Reynold's number at which flow separation occurs is approximately 10 (Example A). Reynold's number is dependent on velocity, size of object, fluid density, and fluid viscosity. A Reynold's number on the order of magnitude of 10 would occur for a low density, low velocity fluid flowing around a small object. Higher velocity flows will separate and behave differently (Examples B – E), but even if an object is to be subjected to this kind of flow behavior, analysis with an ideal flow such as the flow in this program is a reasonable starting point.

The results for the streamlines correlate well with expected streamline results. Figure 1 shows similar streamlines around a circular shape to those produced by the program. Figure 2

shows streamlines around a similarly shaped airfoil that closely replicate those produced by the program. The success of these two cases indicates that the program could successfully analyze idealized flows around other smooth objects.

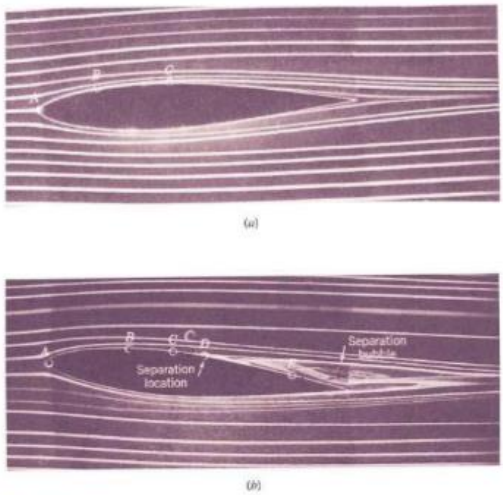


Figure 9.19 Flow visualization photographs of flow past an airfoil (the boundary layer velocity profiles for the points indicated are similar to those indicated in Fig. 9.17*b*): (*a*) zero angle of attack, no separation, (*b*) 5° angle of attack, flow separation. Dye in water. (Photographs courtesy of ONERA, The French Aerospace Lab.)

Figure 2: Airfoil Streamlines with no Separation (Munson et al. p. 514)

The program plots streamlines on two separate occasions: once in a contour plot and once on a streamline plot with animated particles. The contour plot is a plot of the input streamline data from the Excel file, but the streamline plot is calculated using the stream2 function from the horizontal velocity and vertical velocity matrices. The velocity matrices were calculated by differentiating the streamline data. The two streamline plots look almost identical, which indicates that the differentiation was performed correctly.

A major shortcoming of this program is that the results are unitless. The streamlines calculated using Laplace's equation were defined by setting boundary condition values for the streamlines: one at the edge of the boundary and zero at the surface of the object. The boundary conditions make sense because the flow is maximum at the edge of the boundary and zero through the object. However, the result is that the velocity plots only show how the velocity would be distributed around the object without being connected to any units.

APPENDIX 1 – Mathematics Details

- The first operation carried out in this project was solving Laplace's equation numerically in Excel

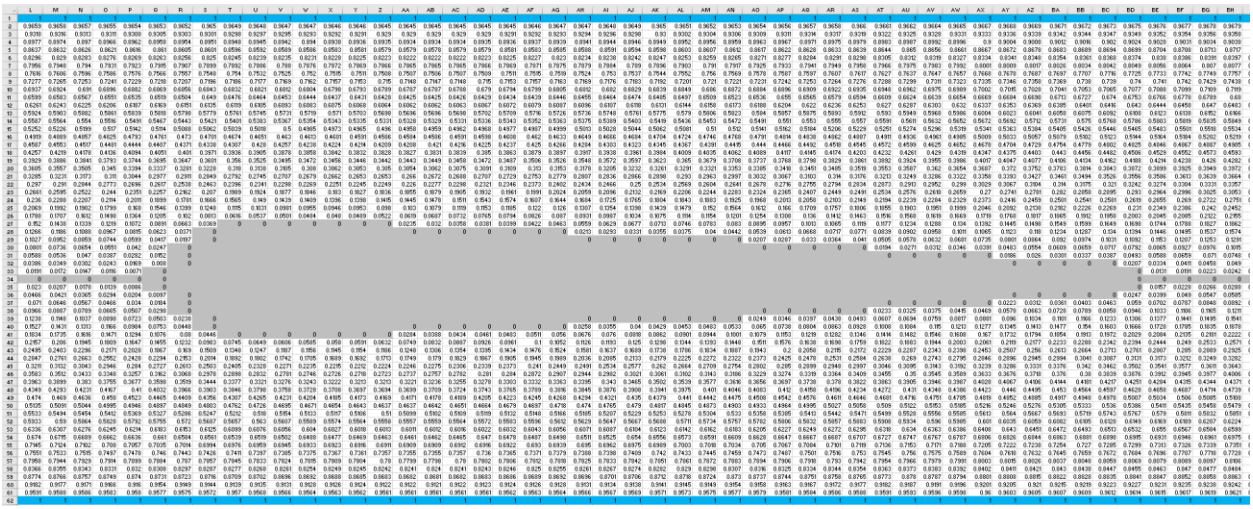


Figure 3: Excel File for Airfoil Streamlines

The boundaries (blue) were set to 1, and the object and the streamline directly through the middle of it were set to zero. The remaining cells were each an average of the surrounding cells, which represents a numerical solution to Laplace's equation:

$$\psi_5 = (\psi_2 + \psi_4 + \psi_6 + \psi_8)/4$$

$$\Psi(x,y) = \frac{f(x+1,y)+f(x-1,y)+f(x,y+1)+f(x,y-1)}{4}$$

Figure 4: Numerical Solution to Laplace's Equation (Munson et al. p740)

- Reynold's Number:

$$\mathrm{Re} = \frac{\rho V \ell}{\mu}$$

ρ – fluid density
μ – dynamic viscosity
V – velocity
l – length of airfoil

- In MATLAB, every entry in the top section of the streamline matrix (psi) was switched from positive to negative in order to maintain the correct sign convention and keep the flow on both sides of the object moving in the same direction. The absolute values of the streamlines were plotted because the contour plot looks cleaner without negative values.

- A fifth degree polynomial surface fit of the streamlines were calculated with a function (createFit) produced by the curve fitting app in MATLAB ("Curve Fitting").

- Goodness of fit statistics:
  - Root mean square error (standard error) of fit ("Evaluating Goodness of Fit"): (values closer to zero indicate a better fit)

    Sum of squares due to error: $SSE = \sum_{i=1}^{n} w_i(y_i - \hat{y}_i)^2$

    Mean square error: $MSE = \frac{SSE}{v}$

    Root mean squared error: $RMSE = s = \sqrt{MSE}$

  - Coefficient of Determination (R^2):

$$r^2 = \frac{S_t - S_r}{S_t} \qquad (14.20)$$

where r^2 is called the *coefficient of determination* and r is the *correlation coefficient* $(= \sqrt{r^2})$. For a perfect fit, $S_r = 0$ and $r^2 = 1$, signifying that the line explains 100% of the variability of the data. For $r^2 = 0$, $S_r = S_t$ and the fit represents no improvement. An alternative formulation for r that is more convenient for computer implementation is

$$r = \frac{n\sum(x_i y_i) - \left(\sum x_i\right)\left(\sum y_i\right)}{\sqrt{n\sum x_i^2 - \left(\sum x_i\right)^2}\sqrt{n\sum y_i^2 - \left(\sum y_i\right)^2}} \qquad (14.21)$$

Figure 5: Coefficient of Determination (Chapra p. 342)

- The horizontal and vertical fluid velocities were calculated by differentiating the streamline data:

Stream function $\qquad u = \frac{\partial \psi}{\partial y} \qquad v = -\frac{\partial \psi}{\partial x}$

Figure 6: Velocities from Stream Function (Munson et al. p. 333)

$$\left(v_x = \frac{d\Psi}{dy} ; v_y = -\frac{d\Psi}{dx} \right)$$

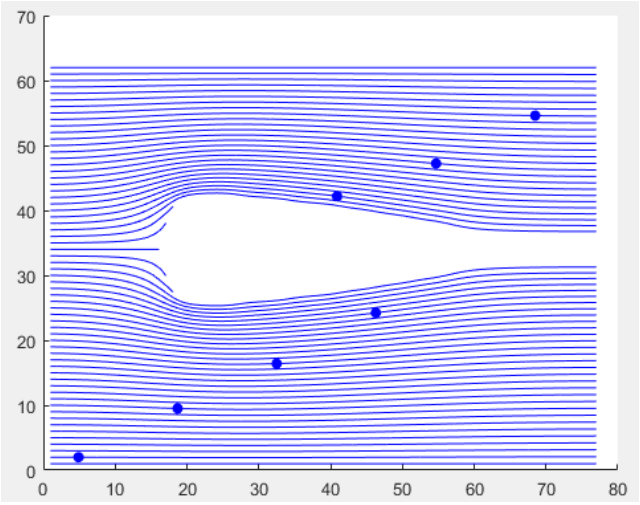
This differentiation was performed in MATLAB using two different functions:

1. gradient – numerical gradient of a matrix $(\frac{\delta}{\delta x}, \frac{\delta}{\delta y})$
2. differentiate – differentiation of surface fit ("Fit Postprocessing")

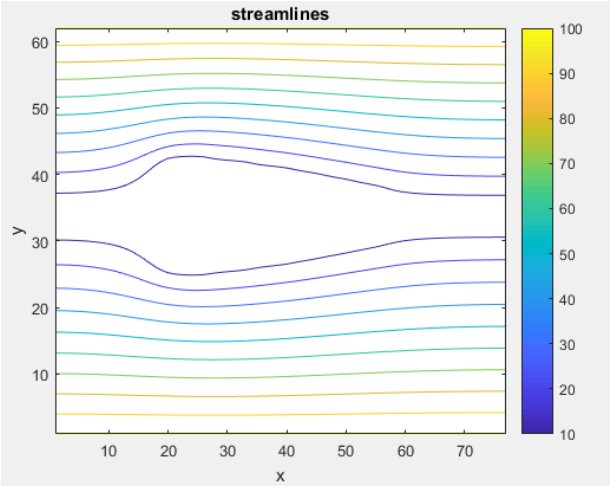
- Calculation of magnitude of velocity from component vectors:

$$V = \sqrt{{v_x}^2 + {v_y}^2}$$

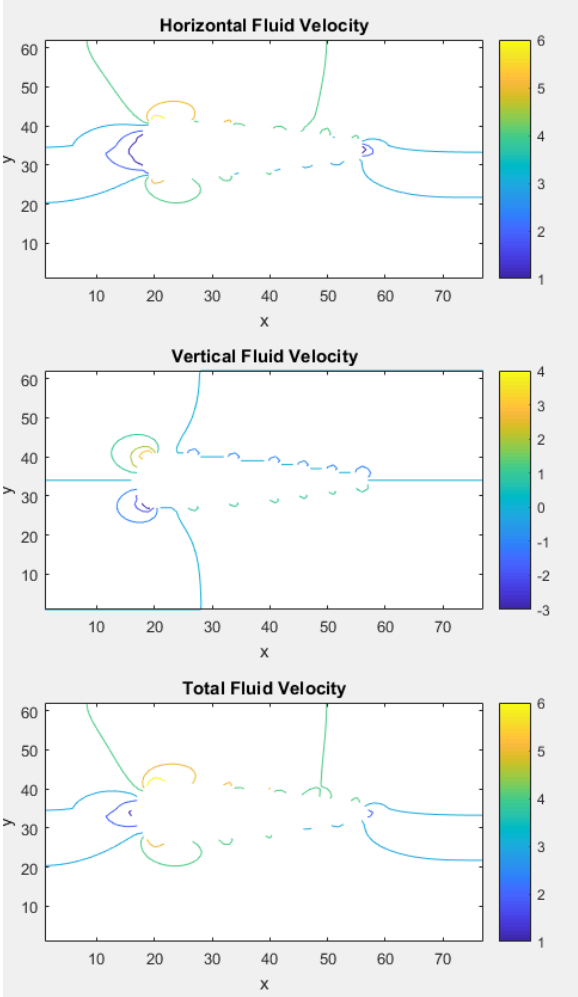
APPENDIX 2 – Sample Output
Airfoil



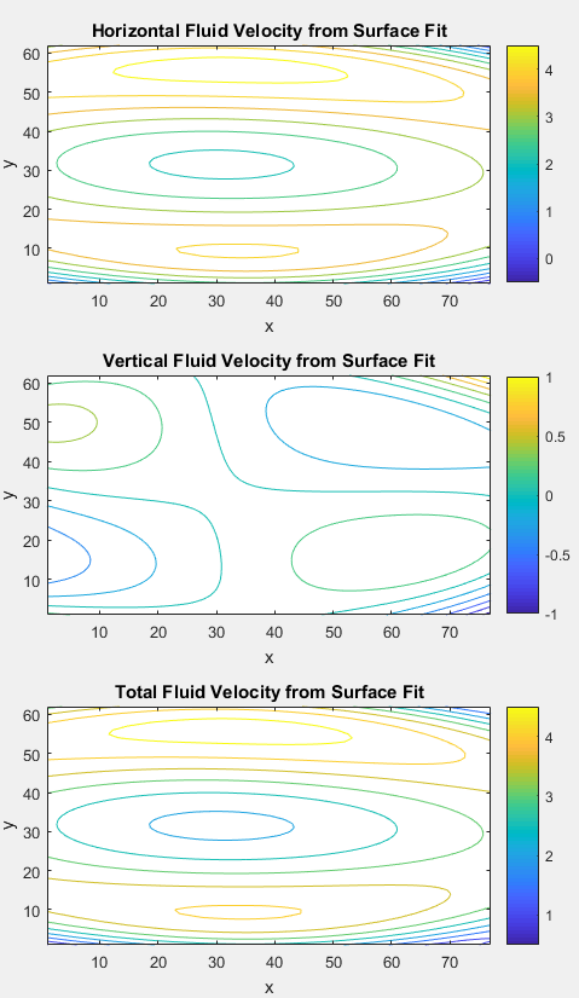
Particle Animation



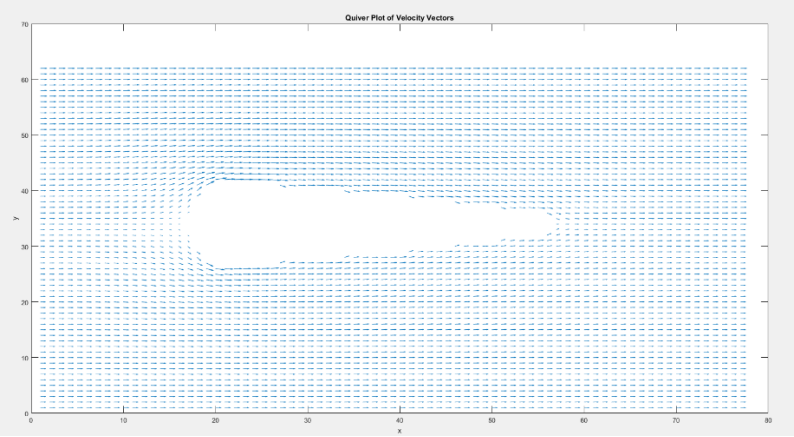
Streamline Contour Plot



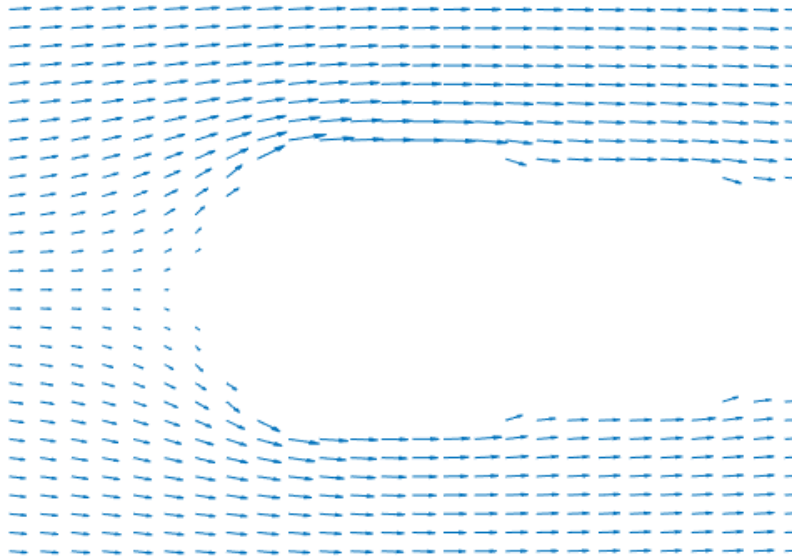
Velocity Plots



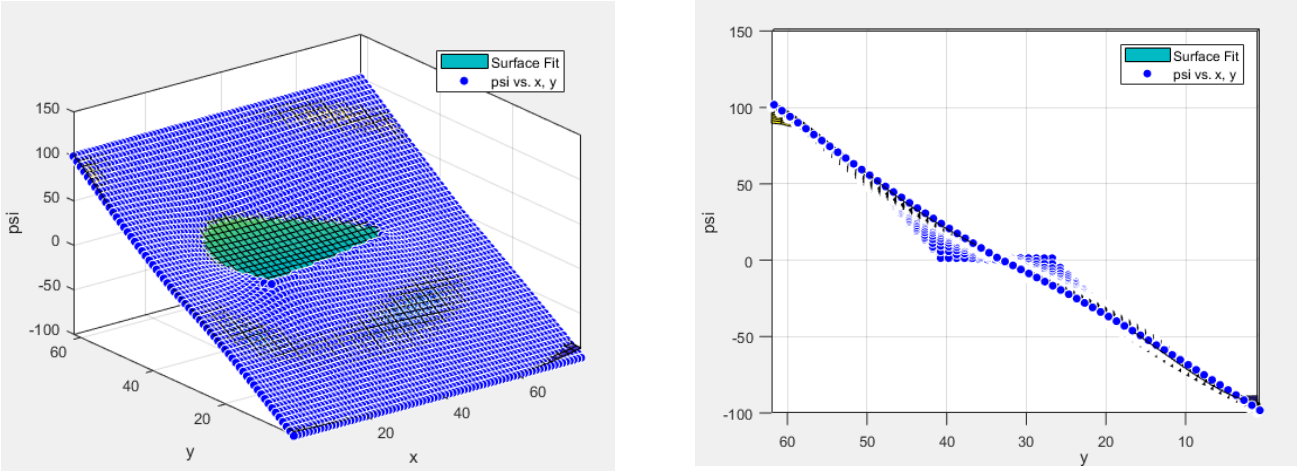
Velocity Plots Derived from Surface Fit



Quiver Plot of Velocity Vectors

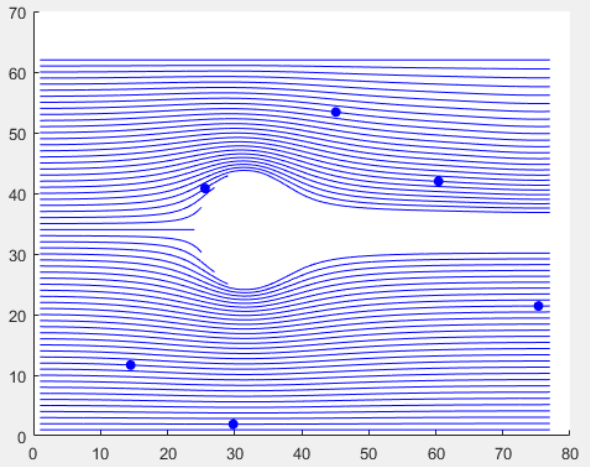


Zoomed View of Quiver Plot

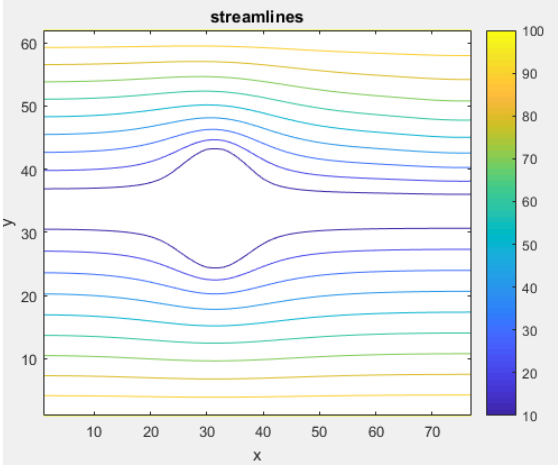


Surface Fit Plots

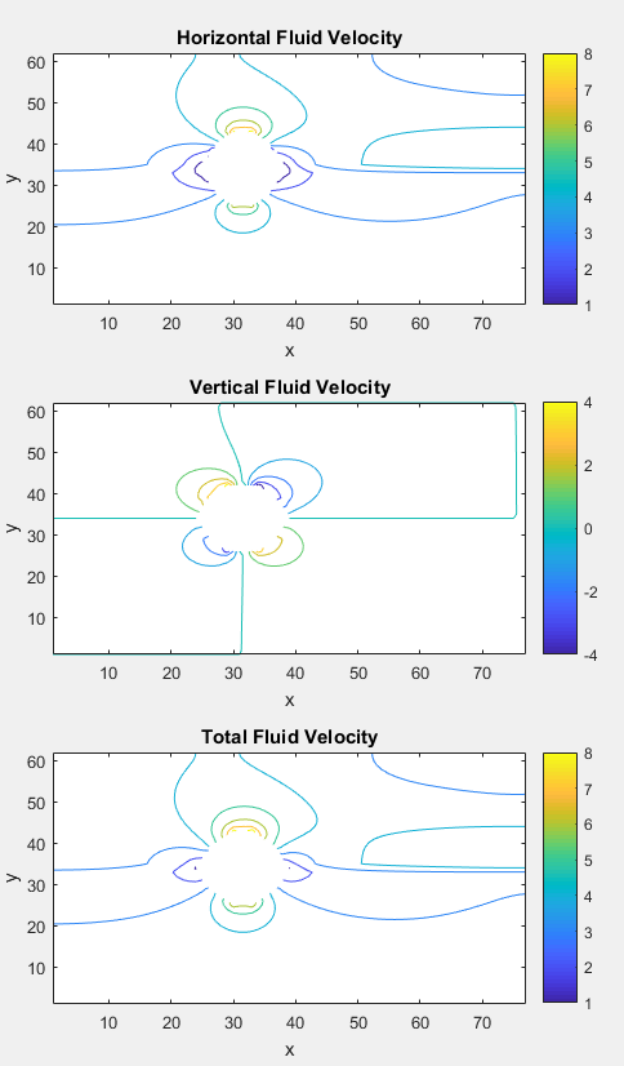
Circle



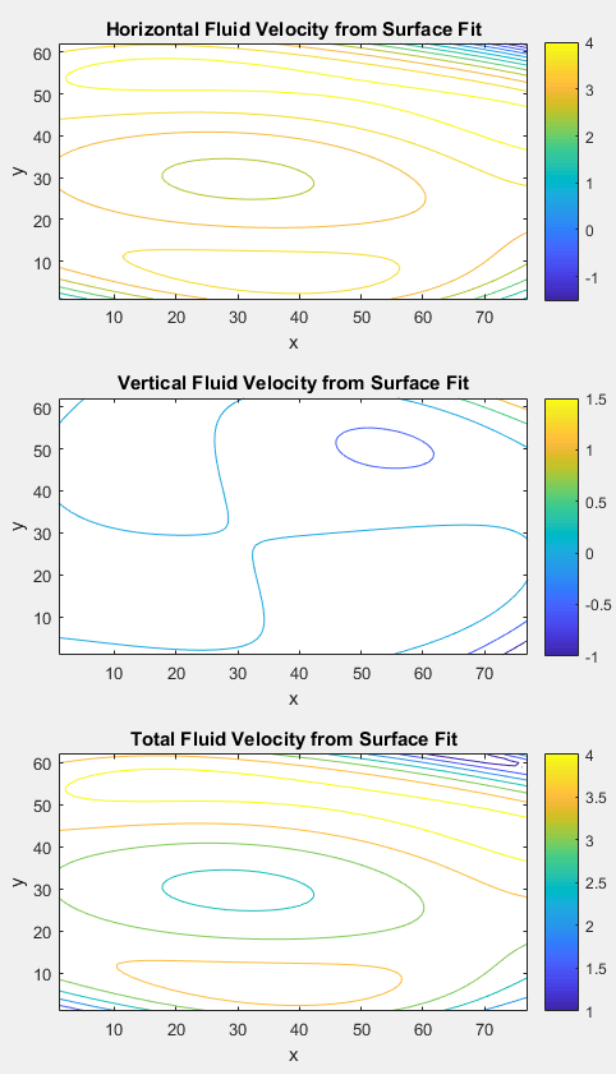
Particle Animation



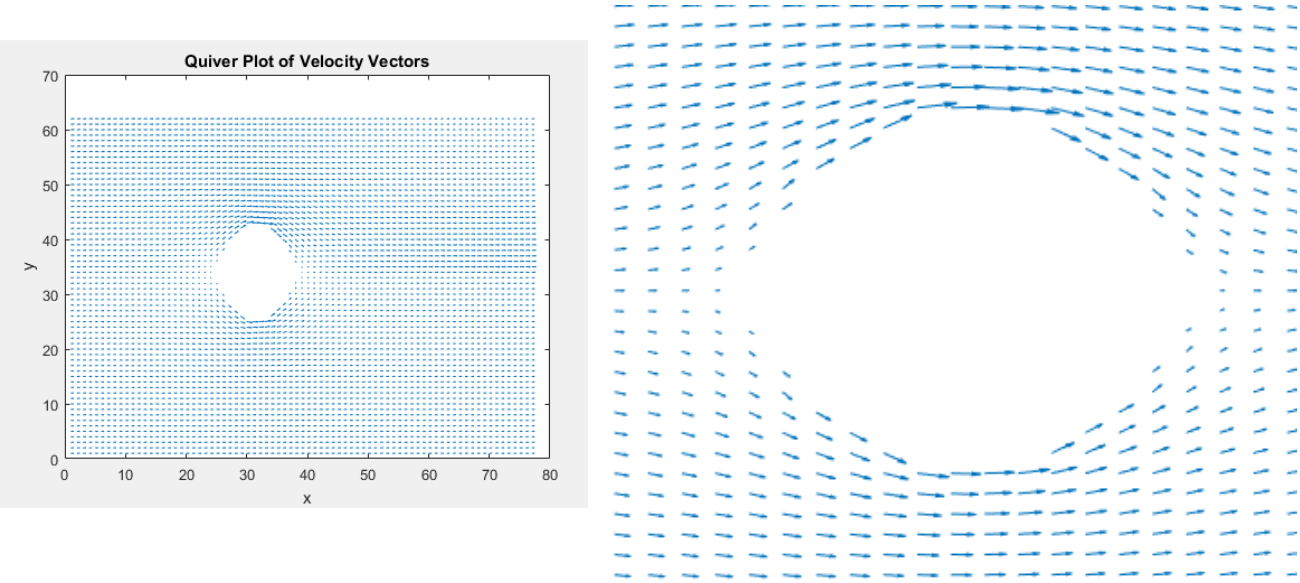
Streamline Contour



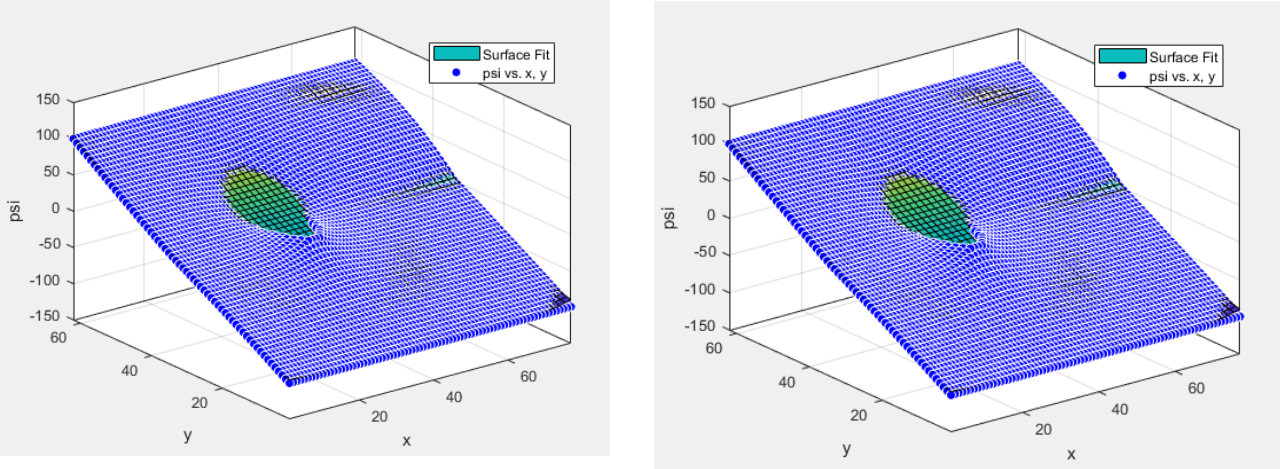
Velocity Plots



Velocity Plots Derived from Surface Fit



Quiver Plot



Surface Fit Plots

APPENDIX 3 – Code

script:

```
% Jim Finnegan
% ES-55 Final Project
% Fluid Flow Around and Airfoil

% choose which shape to analyze
%   options:
%       airfoil (default)
%       circle
sheet = choosesheet;

% read streamline data from excel
psi = xlsread('streamline data', sheet);

% plot streamline absolute values
psi = 100*psi; % amplify for plotting purposes
plotstreamlines(psi)

% adjust psi so velocity can be calculated  with correct sign conventions
%   all values above middle become negative
x = find(psi==0);
for i = 1:x(1)
    psi(i,:) = -psi(i,:);
end

% calculate velocities
%   vx = dPsi/dy
%   vy = -dPsi/dx
%   vtotal = magnitude of component vectors
[m,n] = size(psi); x = 1:n; y = 1:m;
[VY, VX] = gradient(psi);
VY = -VY;
V = sqrt(VX.^2 + VY.^2);

% plot velocities
plotvelocities(VX, VY, V)

% plot quivers
plotquivers(VX,VY)

% surface fit for psi with goodness of fit data
[vyfit, vxfit, vfit, rmse, rsquare] = velocityFit(x, y, psi);

% plot velocities from fit with goodness of fit data
plotfitvelocities(vxfit, vyfit, vfit)

% display goodness of fit data
str = ['Fit Accuracy:' newline '   R^2: ' num2str(rsquare) newline ...
    '   Root mean square error: ' num2str(rmse)];
disp(str);

% plot streamlines and stream particles
animatestreamparticles(VX,VY, x, y)
```

choosesheet

```
function sheet = choosesheet
% sheet = choosesheet
%
% No inputs
%
% Output
%   sheet - user chosen excel sheet to open

% Jim Finnegan
% ES-55 Final Project

% Code adapted from https://www.mathworks.com/help/matlab/ref/dialog.html
%   diaglog that returns output example

%     d = dialog('Position',[300 300 250 150],'Name','Select One');
    d = dialog('Position',[500 500 250 150],'Name','Select One');
    txt = uicontrol('Parent',d,...
           'Style','text',...
           'Position',[20 80 210 40],...
           'String','Select a shape to analyze:');

    popup = uicontrol('Parent',d,...
           'Style','popup',...
           'Position',[75 70 100 25],...
           'String',{'airfoil';'circle'},...
           'Callback',@popup_callback);

    btn = uicontrol('Parent',d,...
           'Position',[89 20 70 25],...
           'String','OK',...
           'Callback','delete(gcf)');

    sheet = 'airfoil';

    % Wait for d to close before running to completion
    uiwait(d);

       function popup_callback(popup,event)
          idx = popup.Value;
          popup_items = popup.String;
          sheet = char(popup_items(idx,:));
       end
end
```

plotstreamlines

```
function plotstreamlines(psi)
% plotstreamlines(psi)
%
% Inputs:
%   psi - streamline data matrix
%
% Outputs:
%   contour plot of streamlines

% Jim Finnegan
% ES-55 Final Project

% plot streamlines
figure
contour(psi); colorbar; title('streamlines'); xlabel('x'); ylabel('y')
```

plotvelocities

```
function plotvelocities(VX, VY, V)
% plotvelocities
%
% Inputs:
%   VX - horizontal fluid velocity matrix
%   VY - vertical fluid velocity matrix
%   V - total fluid velocity matrix
%
% Outputs:
%   Contour plots of each input on a 3x1 subplot figure

% Jim Finnegan
% ES-55 Final Project

% plot velocities
figure
subplot(3,1,1)
contour(VX) % horizontal velocity
colorbar; title('Horizontal Fluid Velocity'); xlabel('x'); ylabel('y')

subplot(3,1,2)
contour(VY) % vertical velocity
colorbar; title('Vertical Fluid Velocity'); xlabel('x'); ylabel('y')

subplot(3,1,3)
contour(V) % total velocity
colorbar; title('Total Fluid Velocity'); xlabel('x'); ylabel('y')
```

plotquivers

```
function plotquivers(VX,VY)
% plotquivers(VX,VY)
%
% Inputs:
%   VX - horizontal fluid velocity matrix
%   VY - vertical fluid velocity matrix
%
% Output:
%   quiver plot of velocity vectors

% Jim Finnegan
% ES-55 Final Project

% plot quivers
figure
quiver(VX,VY)
title('Quiver Plot of Velocity Vectors'); xlabel('x'); ylabel('y')
```

velocityfit

```
function [vyfit, vxfit, vfit, rmse, rsquare, sse] = velocityFit(x, y, psi)
% velocityFit(x, y, psi)
% Surface fit of streamline data to find velocity
%
% Inputs
%   x: input x vector
%   y: inpup y vector
%   psi: streamline matrix
%
% Outputs
%   vyfit: y velocity of fluid
%   vxfit: x velocity of fluid
%   gof: goodness of fit data

% Jim Finnegan
% ES-55 Final Project


% create, plot fit
[fitresult, gof] = createFit(x, y, psi);

% goodness of fit data
rmse = gof.rmse;
rsquare = gof.rsquare;

% calculate velocities by differentiating fit%
%   vx = dPsi/dy
%   vy = -dPsi/dx
%   vtotal = magnitude of component vectors
[xx, yy] = meshgrid(x, y);
[vyfit, vxfit] = differentiate(fitresult, xx, yy);
vyfit = -vyfit;
vfit = sqrt(vyfit.^2 + vxfit.^2);
```

createFit (generated by MATLAB cftool)

```
function [fitresult, gof] = createFit(x, y, psi)
%CREATEFIT(X,Y,PSI)
%  Create a fit.
%
%  Data for 'untitled fit 1' fit:
%      X Input : x
%      Y Input : y
%      Z Output: psi
%  Output:
%      fitresult : a fit object representing the fit.
%      gof : structure with goodness-of fit info.
%
%  See also FIT, CFIT, SFIT.

%  Auto-generated by MATLAB on 07-Dec-2017 17:27:16
%       using 'cftool' app


%% Fit: 'untitled fit 1'.
[xData, yData, zData] = prepareSurfaceData( x, y, psi );

% Set up fittype and options.
ft = fittype( 'poly55' );

% Fit model to data.
[fitresult, gof] = fit( [xData, yData], zData, ft, 'Normalize', 'on' );

% Plot fit with data.
figure( 'Name', 'Surface Fit' );
h = plot( fitresult, [xData, yData], zData );
legend( h, 'Surface Fit', 'psi vs. x, y', 'Location', 'NorthEast' );
% Label axes
xlabel x
ylabel y
zlabel psi
grid on
```

plotfitvelocities

```
function plotfitvelocities(vxfit, vyfit, vfit)
% plotvelocities
%
% Inputs:
%   vxfit - horizontal fluid velocity matrix, differentiated from fit
%   vyfit - vertical fluid velocity matrix, differentiate from fit
%   vfit - total fluid velocity matrix, differentiated from fit
%
% Outputs:
%   Contour plots of each input on a 3x1 subplot figure

% Jim Finnegan
% ES-55 Final Project

% plot velocities from fit with goodness of fit data
figure
subplot(3,1,1)
contour(vxfit); title('Horizontal Fluid Velocity from Surface Fit')
colorbar; xlabel('x'); ylabel('y')

subplot(3,1,2)
contour(vyfit); title('Vertical Fluid Velocity from Surface Fit')
colorbar; xlabel('x'); ylabel('y')

subplot(3,1,3)
contour(vfit); title('Total Fluid Velocity from Surface Fit')
colorbar; xlabel('x'); ylabel('y')
```

animatestreamparticles

```
function animatestreamparticles(VX,VY, x, y)
% animatestreamparticles(VX, VY)
%
% Inputs:
%   VX - horizontal fluid velocity matrix
%   VY - vertical fluid velocity matrix
%
% Output:
%   plot of streamlines, animation of particles traveling on streamlines

% Jim Finnegan
% ES-55 Final Project

% code adapted from: https://www.mathworks.com/help/matlab/ref/...
%                                 streamparticles.html?s_tid=doc_ta
%   streamparticles matlab function description

% and: https://www.mathworks.com/help/matlab/visualize/...
%           creating-stream-particle-animations.html
%   creating stream particle animations

% calculate streamlines
% start streamlines at x=1 for all y values
starty = y;
startx = ones(size(starty));
verts = stream2(x,y,VX,VY,startx,starty);

% plot streamlines and stream particles
figure
streamline(verts)
set(gca,'SortMethod','childorder');
streamparticles(verts,15,...
    'Animate',10,...
    'MarkerFaceColor','blue
```

APPENDIX 4 – References

Chapra, S. C. *Applied Numerical Methods with MATLABÂ® for Engineers and Scientists*. 3rd ed., McGraw-Hill Education, 2012.

"Curve Fitting" *Mathworks.com.* The MathWorks, Inc., 2017. https://www.mathworks.com/help/curvefit/curvefittingapp.html?searchHighlight=cftool&s_tid=doc_srchtitle

"Creating Stream Particle Animations" *Mathworks.com.* The MathWorks, Inc., 2017. https://www.mathworks.com/help/matlab/visualize/creating-stream-particle-animations.html

"dialog" *Mathworks.com.* The MathWorks, Inc., 2017. https://www.mathworks.com/help/matlab/ref/dialog.html?searchHighlight=dialog&s_tid=doc_srchtitle

"Evaluating Goodness of Fit" *Mathworks.com.* The MathWorks, Inc., 2017. https://www.mathworks.com/help/curvefit/evaluating-goodness-of-fit.html#bq_5kwr-3

"Fit Postprocessing" *Mathworks.com.* The MathWorks, Inc., 2017. https://www.mathworks.com/help/curvefit/fit-postprocessing.html

"gradient" *Mathworks.com.* The MathWorks, Inc., 2017. https://www.mathworks.com/help/matlab/ref/gradient.html?searchHighlight=gradient&s_tid=doc_srchtitle

Munson, B. R., et al. *Fundamentals of Fluid Mechanics*. 8th ed., John Wiley & Sons, Inc., 2013.

"stream2" *Mathworks.com.* The MathWorks, Inc., 2017. https://www.mathworks.com/help/matlab/ref/stream2.html

"streamline" *Mathworks.com.* The MathWorks, Inc., 2017. https://www.mathworks.com/help/matlab/ref/streamline.html

"streamparticles" *Mathworks.com.* The MathWorks, Inc., 2017. https://www.mathworks.com/help/matlab/ref/streamparticles.html

"quiver" *Mathworks.com.* The MathWorks, Inc., 2017. https://www.mathworks.com/help/matlab/ref/quiver.html?searchHighlight=quiver&s_tid=doc_srchtitle