THE UNIVERSITY OF
MELBOURNE

MCEN90018: Advanced Fluid Dynamics

Assignment 2

Mischka Kamener, 539 030

Semester 1, 2016

Question 1

The following functions were written to solve a flow over source panels:

```
% MCEN90018: Advanced Fluid Dynamics — Assignment 2
% ------------------------------------------------------------------------
% Mischka Kamener  539030                          Last modified: 28/4/16
%
% Function based of "snippet.m" written by Nick Hutchins. Calculates the
% induced velocities ui and vi in the frame of panel i, by the source
% panel j on the midpoint of panel i.
function [ui, vi] = induced_panel_velocity(xi, yi, xj, yj)

% Find midpoints of panel i and j.
xmi   = mean(xi);
ymi   = mean(yi);
xmj   = mean(xj);
ymj   = mean(yj);

% Calculate distance between mipoints.
r = sqrt((xmj - xmi).^2 + (ymj - ymi).^2);

% Calculate angles.
phi_i =atan2(diff(yi), diff(xi));
phi_j =atan2(diff(yj),diff(xj));
beta = atan2((ymi - ymj),(xmi - xmj));
omega = beta - phi_j;

% Calculate x, y distance between panel midpoints in frame of panel j.
x0_d = r.*cos(omega);
y0_d = r.*sin(omega);
S = sqrt(diff(xj).^2 + diff(yj).^2); % Length of panel j

% Find velocities frame of panel j. q_j = 1, as we take it out of this
% equation.
vj = (1./(2*pi)).*(atan(((S./2)-x0_d)./y0_d)...
    -atan((-(S./2) - x0_d)./y0_d));
uj = (1./(2*pi)).*((-log((y0_d.^2+((S.^2)./4)- (S.*x0_d)+x0_d.^2))./2)...
    + (log((y0_d.^2 + ((S.^2)./4) + (S.*x0_d) + x0_d.^2))./2)); % eqn(29)

% Rotate velocities into frame of panel i.
ui = uj.*cos(phi_j-phi_i)-vj.*sin(phi_j-phi_i);
vi = uj.*sin(phi_j-phi_i)+vj.*cos(phi_j-phi_i);
```

```
% MCEN90018: Advanced Fluid Dynamics — Assignment 2
% ------------------------------------------------------------------------
% Mischka Kamener  539030                          Last modified: 28/4/16
%
% Function based of "example_code_figure5.m" written by Nick Hutchins.
% Calculates the u and v velcoity at locations (xp, yp) due to a source
% panel of strength q placed between (x(1) y(1)) and (x(2) y(2)).
function [u, v] = source_panel_field(q, x, y, xp, yp)

% Calculate midpoint
xm = [mean(x), mean(y)];

% Calculate distances
r = sqrt((xp - xm(1)).^2 + (yp - xm(2)).^2);
S = sqrt(diff(x).^2 + diff(y).^2);

% Calculate angles
phi   = atan2(diff(y), diff(x));
```

```
beta  = atan2(yp - xm(2), xp - xm(1));
omega = beta - phi;

% Calculate distances x0' and y0'
x0_d = r.*cos(omega);   y0_d = r.*sin(omega);

% Velocities in rotated coordinates u' and v'
u_d = (q/(2*pi))*(-0.5*log(y0_d.^2 + S^2/4 - x0_d.*S + x0_d.^2) + ...
    0.5*log(y0_d.^2 + S^2/4 + x0_d.*S + x0_d.^2));
v_d = (q/(2*pi))*(atan((S/2 - x0_d)./y0_d) - atan((-S/2 - x0_d)./y0_d));

% Rotate coodinates back
u   = u_d*cos(phi) - v_d*sin(phi);
v   = v_d*cos(phi) + u_d*sin(phi);
```

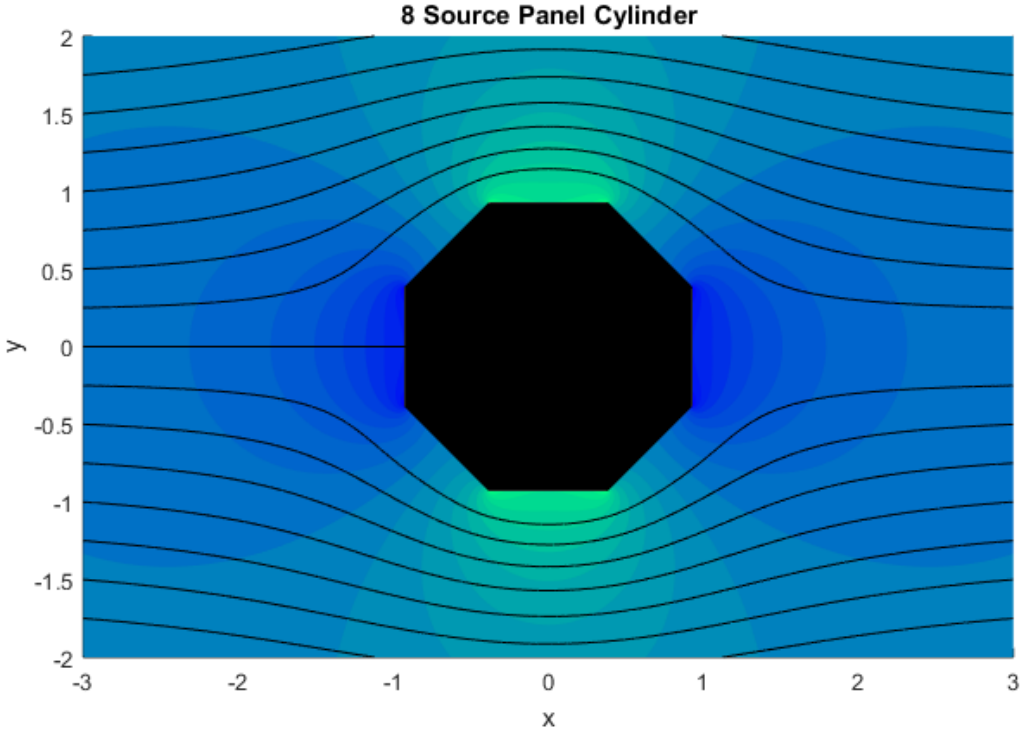
```
% MCEN90018: Advanced Fluid Dynamics - Assignment 2
% ------------------------------------------------------------------------
% Mischka Kamener  539030                          Last modified: 28/4/16
%
% Calculates the source strength q of each of the panels that make up the
% polygon defined in x and y, in a uniform flow U_inf, such that there is
% no flow normal to the midpoint of each panel.
function q = get_source_strengths(x, y, U_inf)

% Get the number of panels from the coordinate matrices
n_panels = length(x) - 1;

% Initialize matrices and vectors
I = 0.5*ones(n_panels); % Initialize to self influence = (1/2)*q
V = zeros(n_panels, 1);

% Loop through all panels
for i = 1:n_panels

    % Set free stream velocity influence
    phi  = atan2(diff(y(i:(i+1))), diff(x(i:(i+1))));
    V(i) =  -U_inf*sin(2*pi - phi);

    % Find influence from all other panels
    for j = 1:n_panels

        % Self influence = (1/2)*q has already been set
        if (i == j), continue; end

        % Find the velocity influence normal to panel i due to panel j.
        [~, I(i,j)] = induced_panel_velocity(x(i:(i+1)), y(i:(i+1))...
            , x(j:(j+1)), y(j:(j+1)));
    end
end

% Solve for source panel strengths
q = I\V;
```

The following script was written to calculate the 8 panel cylinder cross-flow:

```
%% Question 1: Source panel cylinder crossflow
% -- Define constants
n_panels = 8;
r = 1;
U_inf = 1;

% -- Create the panels and grid
% Streamlines go outside of [-2 2] range and return, so create larger grid
```

```
[xp, yp] = meshgrid(-3:.01:3, -3:0.01:3);
panel_sep = 2*pi/n_panels;
theta     = flip((-pi+0.5*panel_sep):panel_sep:(pi+0.5*panel_sep));
x         = r*cos(theta);
y         = r*sin(theta);

% -- Solve for panel strengths and velocity field
q = get_source_strengths(x, y, U_inf);

% Initialsie velocity field
u = U_inf*ones(size(xp));
v = zeros(size(xp));

% Add velocity field contribution from each panel
for i = 1:n_panels
    [ui, vi] = source_panel_field(q(i), x(i:(i+1)), y(i:(i+1)), xp, yp);
    u = u + ui;
    v = v + vi;
end

% -- Plot field
% Find points withing cylinder and set velocity to NaN as area is not
% valid. (Streamlines won't pass through cylinder and contourf won't scale
% based on internal region)
[in, on] = inpolygon(xp,yp,x,y);
u(in & ¬on) = NaN;
v(in & ¬on) = NaN;

% Set the start of streamlines
starty = -2:0.25:2;
startx = -3*ones(size(starty));
```

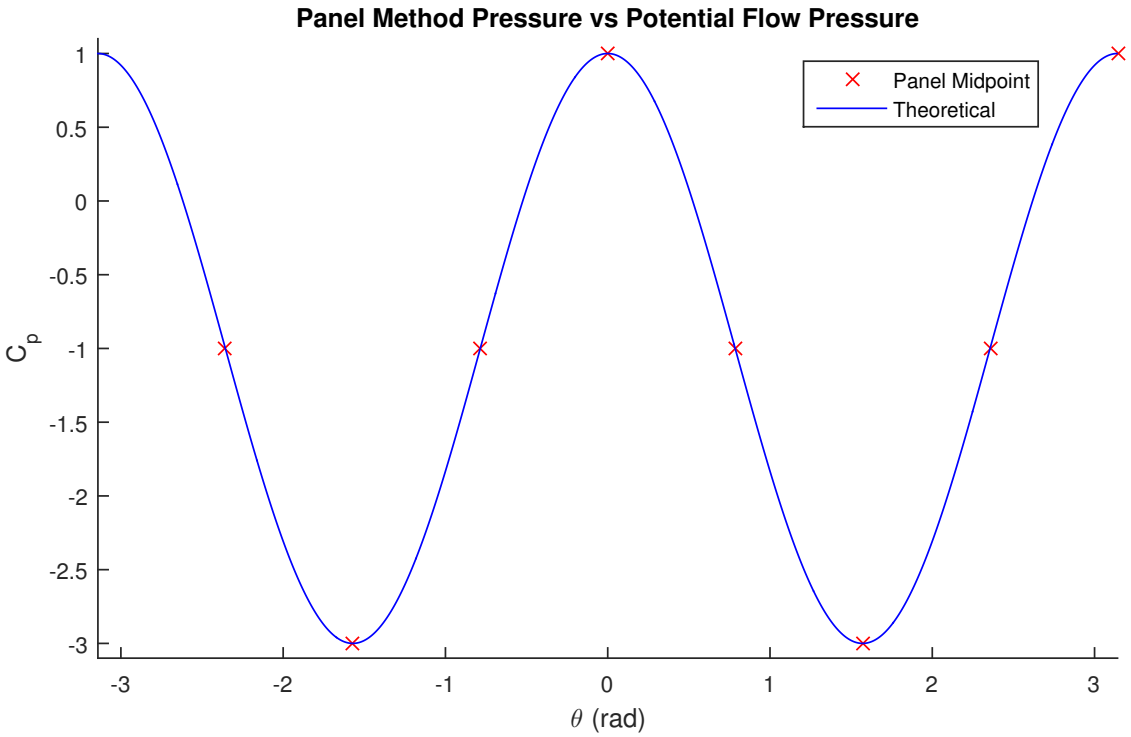
The data was then plotted using the `streamline` function, and using the colormap "winter" for the `contourf` colour plot.



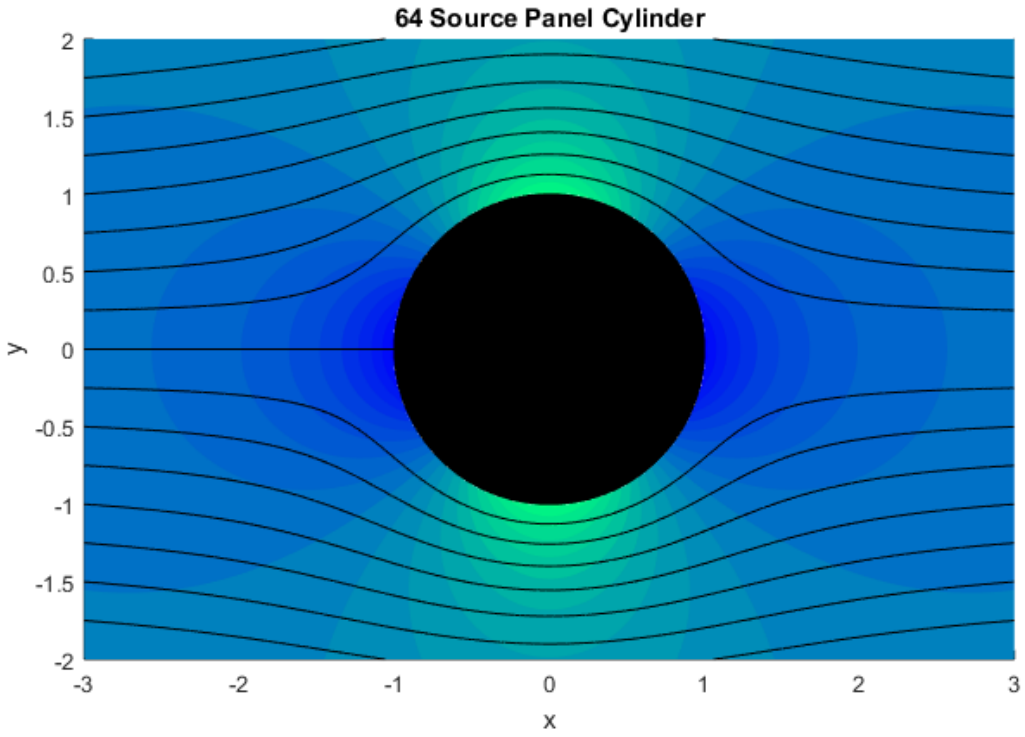
Question 2

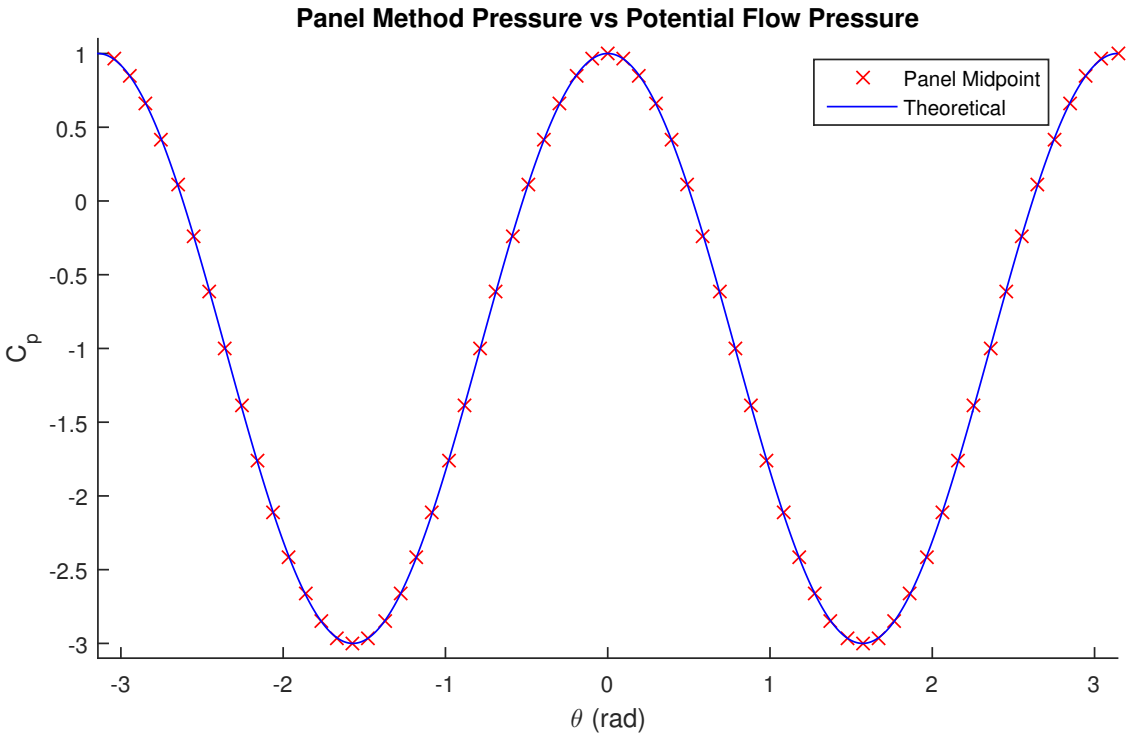
The following code was written to determine the pressure coefficient of the midpoints of the panels, and the pressure distribution around the cylinder according to potential flow. It makes use of some of the variable values from running the code in Question 1.

```
%% Question 2: Potential flow pressure vs panel method
% —— Find pressure at mid—points of panel
Uc = zeros(n_panels, 1);
for i = 1:n_panels

    % Find U_inf contribution
    phi = atan2(diff(y(i:(i+1))), diff(x(i:(i+1))));
    Uc(i) = U_inf*cos(2*pi — phi);

    % Find contributions from other panels
    for j = 1:n_panels
        % Tangential self influence is zero for source panel
        if (i == j), continue; end

        % Set self influence of panel j on panel i;
        [u_i, ¬] = induced_panel_velocity(x(i:(i+1)), y(i:(i+1))...
            , x(j:(j+1)), y(j:(j+1)));
        Uc(i) = Uc(i) + q(j)*u_i;
    end
end
% Calculate pressure coefficient
Cp = 1—(Uc./U_inf).^2;

% Calculate theta at midpoints of panels
thetam = zeros(1, n_panels);
for i = 1:n_panels
    thetam(i) = mean(theta(i:(i+1)));
end

% —— Calculate theoretical pressure
a = 1;
theta_th = —pi:0.01:pi;

% Calculate circle coordinates.
x_th = a*cos(theta_th);  y_th = a*sin(theta_th);

% Calculate velocities from potential flow calcualtions.
u_th = U_inf*(1—a*(x_th.^2—y_th.^2)./(x_th.^2 + y_th.^2).^2);
v_th = —(2*a*U_inf.*x_th.*y_th)./(x_th.^2 + y_th.^2).^2;

% Calculate pressure coefficient.
Cp_th = 1—(sqrt(u_th.^2 + v_th.^2)/U_inf).^2;
```


Panel Method Pressure vs Potential Flow Pressure
Panel Midpoint
Theoretical
C_p
θ (rad)
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
-3
-2
-1
0
1
2
3

Question 3


64 Source Panel Cylinder
y
x
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-3
-2
-1
0
1
2
3



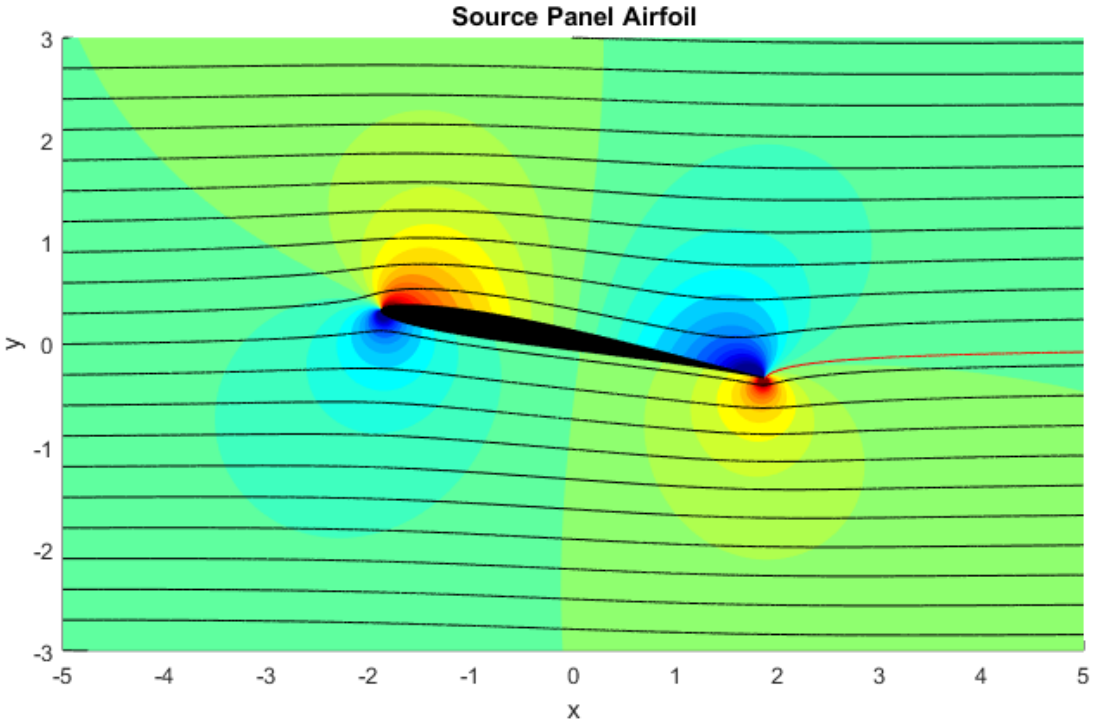
Question 4

The following script was written to find the flow over a Jowkowski airfoil using the source panel method. It makes use of the functions shown in Quesiton 1.

```
%% Question 4: Source panels with Jowkowski airfoil
% —— Create airfoil using Jowkowski transformation
aoa = 10*(pi/180); % Angle of attack (rad)
a = 1;           c = 0.95;
x_s = -0.0498;   y_s = 0.02;
n_panels = 200;
% Streamlines go outside of [-3 3] range and return, so create larger grid.
[xp, yp] = meshgrid(-5:0.01:5, -3.5:0.01:3.5);
U_inf = 1;

% Determine angle shift so that panels start at trailing edge.
shift = atan(y_s./(c-x_s));
theta = flip((0-shift):2*pi/n_panels:(2*pi-shift));

% Transform and rotate coordinates to get airfoil.
z_cs = (a*cos(theta) + x_s) + 1i*(a*sin(theta) + y_s);
z_j  = exp(-1i*aoa)*(z_cs + (c^2./z_cs));
x = real(z_j);
y = imag(z_j);

% —— Use source panel method to get flow around airfoil.
% Solve for panel strengths
q = get_source_strengths(x, y, U_inf);

% Initialsie velocity field
u = U_inf*ones(size(xp));
v = zeros(size(xp));

% Add velocity field contribution from each panel
```

```
for i = 1:n_panels
    [ui, vi] = source_panel_field(q(i), x(i:(i+1)), y(i:(i+1)), xp, yp);
    u = u + ui;
    v = v + vi;
end

% Discard velocities within airfoil.
[in, on] = inpolygon(xp,yp,x,y);
u(in & ¬on) = NaN;
v(in & ¬on) = NaN;

% Set start of streamlines, streamline from trailing edge starts at x(1)
% y(1) and is added individually in the plot.
starty = -3:0.3:3;
startx = -5*ones(size(starty));
```

The streamlines are plotted using the **`streamline`** function. Note that the Kutta condition is not met, as the source panels do not add the required circulation.



Question 5

The source panel functions from Question 1 were modified to vortex panel functions. There are given below:

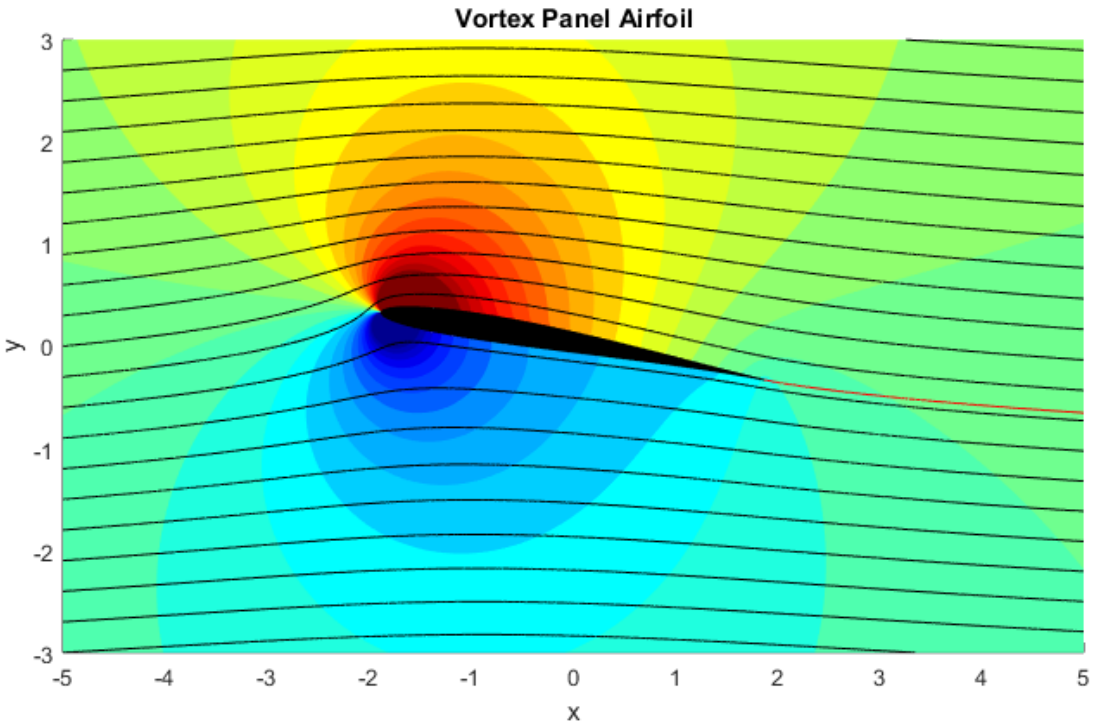
```
% MCEN90018: Advanced Fluid Dynamics — Assignment 2
% ————————————————————————————————————————————————————————————
% Mischka Kamener  539030                              Last modified: 28/4/16
%
% Function based of "snippet.m" written by Nick Hutchins. Calculates the
% streamfunction by the vortex panel j on the midpoint of panel i.
function psi_i = induced_panel_streamfunction(xi, yi, xj, yj)

% Find midpoints of panel i and j.
xmi    = mean(xi);
ymi    = mean(yi);
xmj    = mean(xj);
ymj    = mean(yj);

% Calculate distance between mipoints.
r = sqrt((xmj — xmi).^2 + (ymj — ymi).^2);

% Calculate angles.
phi_j =atan2(diff(yj),diff(xj));
beta = atan2((ymi — ymj),(xmi — xmj));
omega = beta — phi_j;

% Calculate x, y distance between panel midpoints in frame of panel j.
x0_d = r.*cos(omega);
y0_d = r.*sin(omega);
S = sqrt(diff(xj).^2 + diff(yj).^2); % Length of panel j.
a = —S/2;
b = S/2;

% Calculate streamfunction psi at the midpoint of panel i. Do not need to
% change coordinates, as streamfunction is a scalar. gamma_j = 1, as we
% take it out of this equation.
psi_i = (1./(2.*pi)).*((((x0_d—b)./2).*log((x0_d—b).^2 + y0_d.^2) + ...
    y0_d.*atan((x0_d—b)./y0_d) + b) — ...
    (((x0_d—a)./2).*log((x0_d—a).^2+y0_d.^2) + ...
    y0_d.*atan((x0_d—a)./y0_d) + a));
```

```
% MCEN90018: Advanced Fluid Dynamics — Assignment 2
% ————————————————————————————————————————————————————————————
% Mischka Kamener  539030                              Last modified: 28/4/16
%
% Function based of "example_code_figure5.m" written by Nick Hutchins.
% Calculates the u and v velcoity at locations (xp, yp) due to a vortex
% panel with circulation gamma placed between (x(1) y(1)) and (x(2) y(2)).
function [u, v] = vortex_panel_field(gamma, x, y, xp, yp)

% Calculate midpoint
xm = [mean(x), mean(y)];

% Calculate distances
r = sqrt((xp — xm(1)).^2 + (yp — xm(2)).^2);
S = sqrt(diff(x).^2 + diff(y).^2);

% Calculate angles
phi    = atan2(diff(y), diff(x));
beta   = atan2(yp — xm(2), xp — xm(1));
omega = beta — phi;
```

```
21
22 % Calculate distances x0' and y0'
23 x0_d = r.*cos(omega);   y0_d = r.*sin(omega);
24
25 % Velocities in rotated coordinates u' and v'
26 v_d = (gamma/(2*pi))*(-0.5*log(y0_d.^2 + S^2/4 - x0_d.*S + x0_d.^2) + ...
27     0.5*log(y0_d.^2 + S^2/4 + x0_d.*S + x0_d.^2));
28 u_d = -(gamma/(2*pi))*(atan((S/2 - x0_d)./y0_d) - ...
29     atan((-S/2 - x0_d)./y0_d));
30
31 % Rotate coodinates back
32 u   = u_d*cos(phi) - v_d*sin(phi);
33 v   = v_d*cos(phi) + u_d*sin(phi);
```

```
1  % MCEN90018: Advanced Fluid Dynamics - Assignment 2
2  % ------------------------------------------------------------------------
3  % Mischka Kamener  539030                          Last modified: 28/4/16
4  %
5  % Calculates the circulation (gamma) of each of the vortex panels that make
6  % up the airfoil defined in x and y, in a uniform flow [U_inf, V_inf], such
7  % that the midpoints of each panel lie along a streamline. Additionaly, the
8  % first panel and last panel are set to have the same circulation, so that
9  % the Kutta condition is met.
10 function gamma = get_vortex_strengths(x, y, U_inf, V_inf)
11
12 % Get the number of panels from the coordinate matrices.
13 n_panels = length(x) - 1;
14
15 % Initialize matrices and vectors.
16 I = zeros(n_panels + 1);
17 V = zeros(n_panels + 1, 1);
18
19 % Set final row of I to satisfy Kutta condition, and final column to
20 % satisfy constant streamfunction condition.
21 I(end,1) = 1;           % Kutta
22 I(end, end-1) = 1;      % Kutta
23 I(1:(end-1),end) = 1;   % Constant streamfunction
24
25 % Loop through all panels
26 for i = 1:n_panels
27
28     % Set free stream streamfunction influence
29     xm = mean(x(i:(i+1)));
30     ym = mean(y(i:(i+1)));
31     V(i) =  -U_inf*ym + V_inf*xm;
32
33     % Find influence from all other panels
34     for j = 1:n_panels
35
36         % Set influence of panel j on midpoint of panel i;
37         I(i,j) = induced_panel_streamfunction( ...
38             x(i:(i+1)), y(i:(i+1)), x(j:(j+1)), y(j:(j+1)));
39     end
40 end
41 % Solve for source panel strengths. Discard the streamfunction constant.
42 gamma = I\V;
43 gamma = gamma(1:(end-1));
```

The script to determine the flow over the airfoil is exactly the same in Question 4, except the function calls are replaced. `get_vortex_strengths` is called instead of `get_source_strengths`, and `vortex_panel_field` is called instead of `source_panel_field`. Additionally, only 64 panels are used, as much fewer panels are required to get a good result.



Question 6

A vortex panel method solution to the flow over an airfoil is based off potential flow theory, and therefore has the same limitations. Potential flow assumes incompressible flow, inviscid flow, and has no vorticity. This means that the model has no boundary layer, and the no slip condition is not met, and there is no turbulence. No boundary layer means that there is no skin friction, only pressure drag; and there is no flow separation, so the airfoil does not stall at high angles of attack. The incompressible assumption also restricts the velocity range where the model is applicable.

Because of these limitations, the vortex panel method only gives a good approximation of the lift and drag of an airfoil when the following criteria are met:

- Low thickness to chord ratio (as separation is not modelled)
- Low angles of attack (as separation is not modelled)
- Low velocity (as flow is assumed to be incompressible). Less than roughly $0.3 \times M$ is a reasonable approximation.
- The oncoming flow is laminar (as turbulence is not modelled)

Additionally, since only pressure drag (no skin friction) can be determined using panel methods, the accuracy of the approximation of the total drag increases with higher Reynold's number. This is because high Reynolds number means that the inertial forces are much larger than the viscous forces, so pressure drag becomes much larger than drag due to skin friction. Therefore ignoring skin friction at higher Reynolds number creates only a relatively small error.

Question 7

The following script written to calculate the potential flow solution to the streamlines over the Jowkowski airfoil.

```
%% Question 7: Plot potential flow streamlines
% Determine starting points in untransformed plane
aoa = 10*(pi/180); % Angle of attack (rad)
a = 1;          c = 0.95;
x_s = -0.0498;  y_s = 0.02;

% Find potential flow solution to flow over cylinder.
U_inf = 1;
Gamma = 4*pi*U_inf*a*sin(asin(y_s/a)+aoa);
[x,y] = meshgrid(-6:0.01:12,-6:0.01:6);
z = x +1i.*y;
% Complex potential function for flow over cylinder is given by:
% w = U_inf*(z+a.^2./z)+(1i*Gamma./(2*pi)).*log(z./a);
dwdz = -a.^2*U_inf./(z.^2)+U_inf+1i*Gamma./(2*pi.*z);
u =  real(dwdz);
v = -imag(dwdz);

% -- Find streamlines for flow over cylinder and use Jowkowski transform to
%    get back to flow over airfoil.
% Set start locations based of Q5 specification, and reverse Jowkowski
% transform.
starty = -3:0.3:3;    startx = -5*ones(size(starty));
startz = startx + 1i*starty;
startz = [startz (1.8712 - 1i*0.32996)];

startz = startz*exp(1i*aoa);
 % Choose branch that gives solutions outside of cylider.
startz = 0.5*(startz-sqrt(startz.^2-4*c));
startz = startz - (x_s + 1i*y_s);
startz = startz*exp(-1i*aoa);

% Find cylinder boundary, and discard velocity values that lie within.
theta = 0:0.01:2*pi;
x_circ = a*cos(theta);
y_circ = a*sin(theta);
z_circ = x_circ + 1i*y_circ;
[in, on] = inpolygon(x,y,x_circ,y_circ);
u(in & ~on) = NaN;
v(in & ~on) = NaN;

% Find streamlines around cylinder, and convert to complex coordinates.
xy = stream2(x, y, u, v, real(startz), imag(startz));
z_pflow = zeros(10000, size(xy,2));
for i = 1:size(xy,2)
    xy_i = cell2mat(xy(i));
    z_pflow(:,i) = xy_i(:,1) + 1i*xy_i(:,2);
end

% Jowkowski transform the coordinates of the streamlines to get streamlines
% over airfoil.
z_pflow = z_pflow*exp(1i*aoa);
z_pflow = z_pflow + (x_s + 1i*y_s);
z_pflow = z_pflow + c^2./(z_pflow);
z_pflow = z_pflow*exp(-1i*aoa);
```

The streamlines are compared against the panel method streamlines in the figure below. As can be seen, the streamlines from the two methods are almost identical.