Assignment - 3

1 Stream function-vorticity formulation of Navier-Stokes equation

The vorticity-stream function approach has been one of the most popular methods for solving the 2-D incompressible Navier-Stokes equations. In this approach, a change of variables is made that replaces the velocity components with the vorticity ω and the stream function ψ. The vorticity vector ω is defined as

$$\omega = \nabla \times \mathbf{V}$$

where $\mathbf{V}$ is the velocity vector.

In 2-D, the vorticity vector has only one component, normal to the 2-D plane

$$\omega = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}$$

where u and v the x and y components of the velocity vector.

The streamfunction is defined as

$$\begin{aligned} u &= \frac{\partial \psi}{\partial y} \\ v &= -\frac{\partial \psi}{\partial x} \end{aligned}$$

The definition of ψ identically satisfies the incompressible continuity equation (check!)

The two momentum equations can be combined by eliminating pressure to get (take $\nabla\times$ of the momentum equations; $\partial/\partial y$ of first equation and $\partial/\partial x$ of the second equation and subtract)

$$\frac{\partial \omega}{\partial t} + u\frac{\partial \omega}{\partial x} + v\frac{\partial \omega}{\partial y} = \frac{1}{Re}\left(\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2}\right) \quad (1)$$

This is vorticity transport equation in 2D (VTE).

Substituting the definition of u and v in terms of ψ into the definition of ω, we get

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = -\omega$$

As a result of the change of variables, we have been able to separate the mixed elliptic-parabolic 2-D incompressible Navier-Stokes equations into one parabolic equation (the vorticity transport equation) and one elliptic equation (the Poisson equation). These equations are normally solved sequentially using a time-marching procedure, which is described by the following steps:

1. Specify initial values for ω and ψ at time $t = 0$. (Take $\omega = 0$ and $\psi = 0$)
2. Iterate for new ψ values at all points by solving the Poisson equation using new ω at interior points.
3. Determine values of ω on the boundaries using ψ and ω values at interior points.
4. Solve the vorticity transport equation for ω at each interior grid point at time $t + \Delta t$
5. Find the velocity components
6. Return to Step 2

2 Discretization

1. Use second-order central differencing for Poisson's equation.
2. Use an explicit time difference scheme for VTE
3. Use second-order central differencing for diffusion term in VTE.
4. For convection terms use upwind scheme
   (a) The first derivative can be written as
   $$\frac{\partial u}{\partial x} = \frac{u_{i+1} - u_{i-1}}{2\Delta x} + \alpha \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2}$$
   i.e., central differencing plus second-order artificial diffusion
   (b) $\alpha = -\Delta x/2$ makes it
   $$\frac{\partial u}{\partial x} = \frac{u_i - u_{i-1}}{\Delta x}$$

(c) $\alpha = \Delta x/2$ makes it

$$\frac{\partial u}{\partial x} = \frac{u_{i+1} - u_i}{\Delta x}$$

(d) Then the first order upwind for convection can be written as

$$a\frac{\partial u}{\partial x} = a\frac{u_{i+1} - u_{i-1}}{2\Delta x} - |a|\frac{u_{i+1} - 2u_i + u_{i-1}}{2\Delta x}$$

So, depending on the sign of a, the above expression gives an appropriate upwind scheme

(e) The same approach can be used to get a higher order expression for $a\frac{\partial u}{\partial x}$

$$a\frac{\partial u}{\partial x} = a\frac{-u_{i+2} + 8(u_{i+1} - u_{i-1}) + u_{i-2}}{12\Delta x} + |a|\frac{u_{i+2} - 4u_{i+1} + 6u_i - 4u_{i-1} + u_{i-2}}{12\Delta x}$$

This is 4th order central difference plus fourth order dissipation. Please check. This gives an appropriate 3rd order upwind scheme based on the sign of a.

(f) When using a higher order scheme, the points near the boundary need to be discretized using a lower order scheme as you would be forced to take a smaller stencil.

3 Boundary Condition

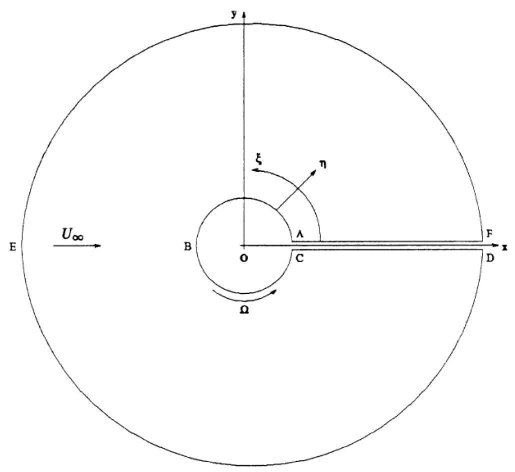
1. Refer the figure

2. ABC is the cylinder, FED is the outer boundary, AF and CD are the periodic boundary condition

3. Since it is a stationary cylinder, the value of ψ on the cylinder is constant - no penetration boundary condition. Take this value to be $\psi_{cylinder} = constant = 20$

4. On the outer boundary, define $\psi = V_\infty y + constant(i.e.\ 20)$, where y is the y coordinate of the point on the boundary.

5. On the outer boundary, either take $\omega = 0$ or impose $\frac{\partial \omega}{\partial r} = 0$



6. On the wall, use

$$\omega_{i,1} = \frac{2(\psi_{i,1} - \psi_{i,2})}{\Delta r^2}$$

where in (i, j) the first index varies along the wall and the second normal to the wall. Then, index i varies along the wall and index 1 stands for point on the wall.

4 Problem

1. Solve viscous flow around a circular cylinder at a Reynolds number of 200.

   (a) Consider a circular cylinder of diameter d_1=1.

   (b) Take the outer boundary to be at a diameter of $20 * d_1$

   (c) Generate a grid around the cylinder; take uniform distribution in θ and in radial direction. Take 121 divisions in theta.

   (d) Solve the equation in cylindrical coordinates.

   (e) Since the domain is multiply connected, take a cut line downstream of the cylinder at $y = 0$ in the domain (AF and CD in the figure). This makes the domain 4 sided.

   (f) Apply periodic boundary condition on this boundary.

(a) Use the following methods to solve the equation

    i. Gauss-Seidel method with relaxation

(b) Since you are solving unsteady flow there would vortex shedding from the cylinder. Show the results.

Assignment no. 3

Stream function - vorticity formulation of Navier - Stokes equation

Computational Methods for Compressible Flows
Date – 12/04/18

Vasu Bansode
SC15B009
Department of Aerospace engineering
Indian Institute of Space Science and Technology, Thiruvananthapuram-695547

Abstract –

Computational fluid dynamics provide efficient way to solve complex flow problems. Here, 2-D incompressible Navier Stokes equation for flow over a rectangular cylinder is solved using Gauss Seidel method with relaxation as an iterative method for Re = 200. Various plots are displayed at the end of the report.

Theory –

The vorticity-stream function approach has been one of the most popular methods for solving the 2-D incompressible Navier-Stokes equations. In this approach, a change of variables is made that replaces the velocity components with the vorticity ω and the stream function ψ . The vorticity vector ω is defined as –

$$\boldsymbol{\omega} = \boldsymbol{\nabla} \times \boldsymbol{V}$$

Where, $\boldsymbol{V}$ is the velocity vector. In 2-D case $\boldsymbol{\omega}$ has only one component which is perpendicular to the *xy* plane.

$$\omega = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}$$

Where, *u* and *v* are *x* and *y* components of velocity.

The stream function is defined as,

$$u = \frac{\partial \psi}{\partial y}$$

$$v = -\frac{\partial \psi}{\partial x}$$

The definition of ψ identically satisfies the incompressible continuity equation.

The two momentum equations can be combined by eliminating pressure to get (take $\boldsymbol{\nabla} \times$ of the momentum equations; $\frac{\partial}{\partial y}$ of 1st equation and $\frac{\partial}{\partial x}$ of the 2nd equation and subtract)

$$\frac{\partial \omega}{\partial t} + u\frac{\partial \omega}{\partial x} + v\frac{\partial \omega}{\partial y} = \frac{1}{Re}\left(\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2}\right)$$

This is vorticity transport equation in 2D (VTE).
Substituting the definition of *u* and *v* in terms of ψ into the definition of ω, we get –

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = -\omega$$

As a result of the change of variables, we have been able to separate the mixed elliptic-parabolic 2-D incompressible Navier-Stokes equations into one parabolic equation (the vorticity transport equation) and one elliptic equation (the Poisson equation). These equations are normally solved sequentially using a time-marching procedure.

Variables –

1. Diameter of cylinder = 1 m
2. Free stream velocity = 1 m/s

Boundary Conditions-

1. Physical domain is from 0.5 m to 10 m.
2. No penetration boundary condition on the cylinder surface i.e. ψ = *constant = 20*
3. On the outer boundary we have $\psi = V_{inf}\, y + 20$ where *y* is the co-ordinate of the point on the boundary.
4. On the outer boundary $\omega = 0$.
5. On the cylinder wall

$$\omega = \frac{2(\psi_{i,1} - \psi_{i,2})}{dr^2}$$

Where in *(i, j)* the 1st index varies along the wall and the second normal to the wall. Then, index *i* varies along the wall and index *j* stands for point on the wall.

6. Periodic boundary conditions for both ω and ψ at $y = 0$ since the domain is multiply connected.

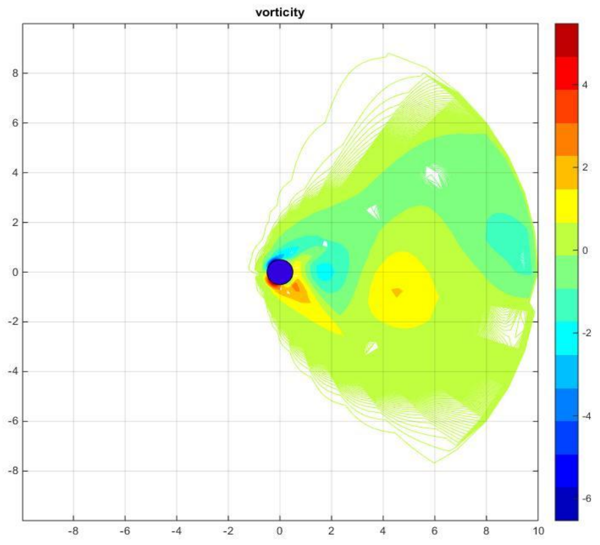
Problem Scheme-

1. The problem is solved in the cylindrical co-ordinates.
2. Radial distance is discretised into 40 elements.
3. Angular position is varied from 0 degree to 360 degree with 40 elements.
4. Time is discretised into 1000 elements from 0 to 100 sec.
5. Total residue is set as convergence criteria.
6. Convergence is achieved when total residue goes beyond 0.01.
7. For Gauss Seidel method with relaxation, relaxation factor is equal to 1.93.
8. Second-order central differencing for Poisson's equation.
9. An explicit time difference scheme for VTE.
10. For convection terms use upwind scheme.

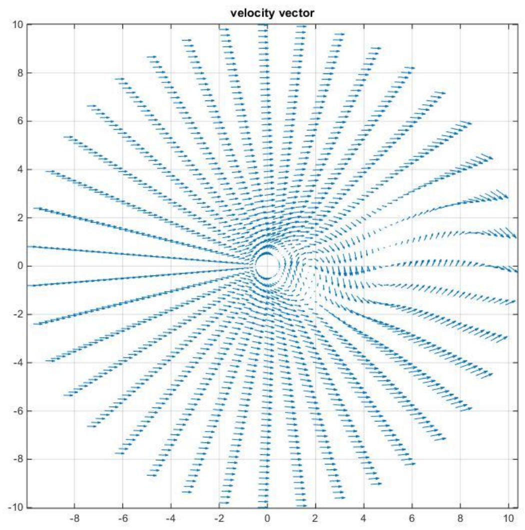
Results –

Results are shown at the end of 100 seconds.

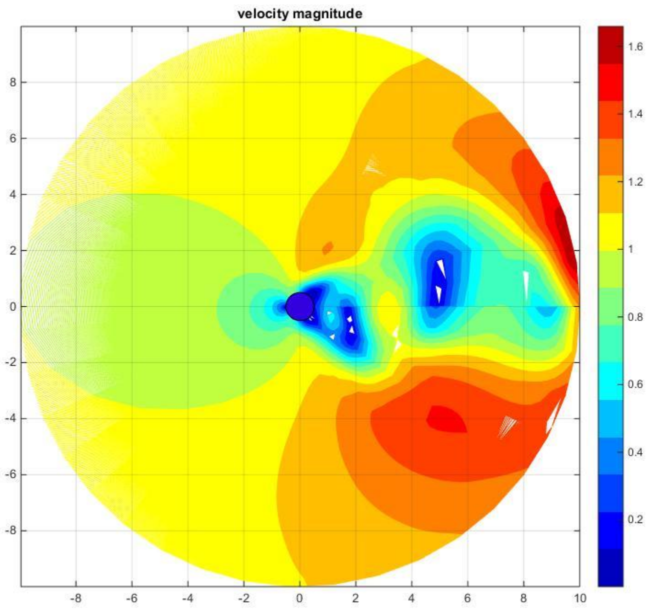
1. **Vorticity –**



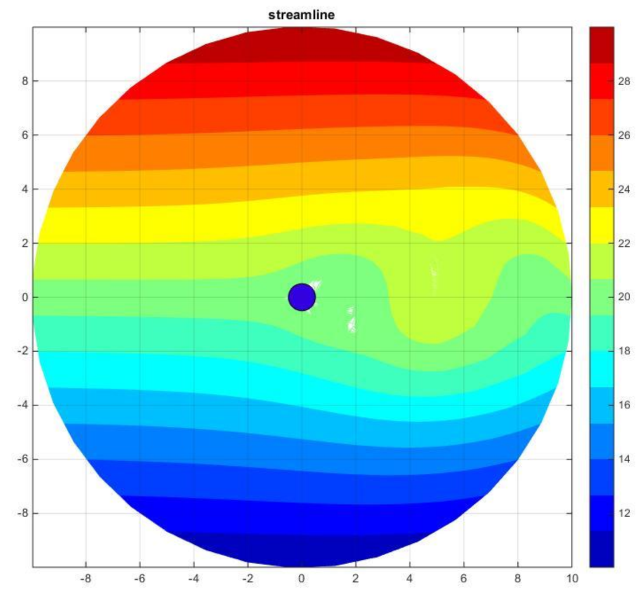
2. Velocity vector –



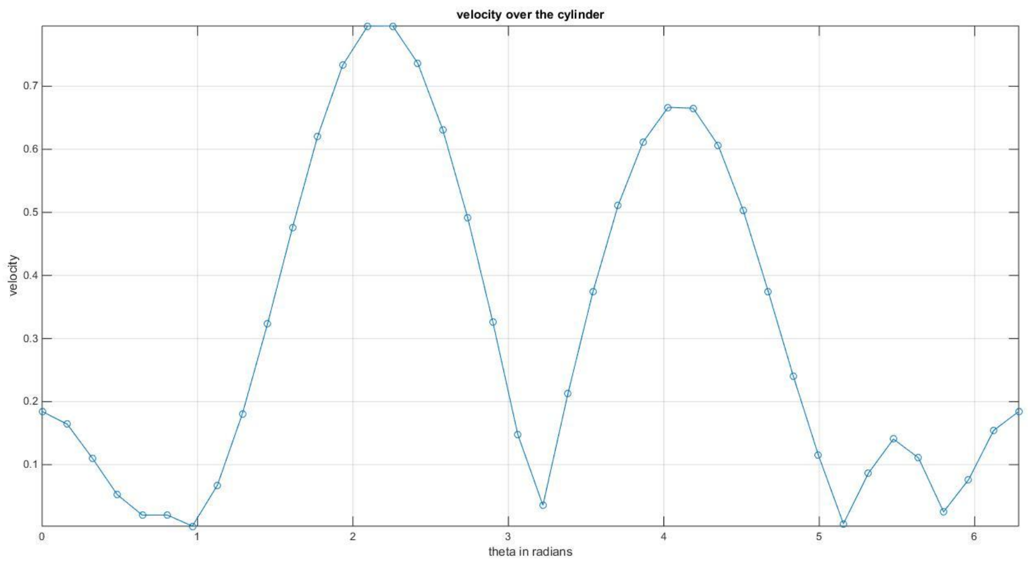
3. Velocity magnitude –



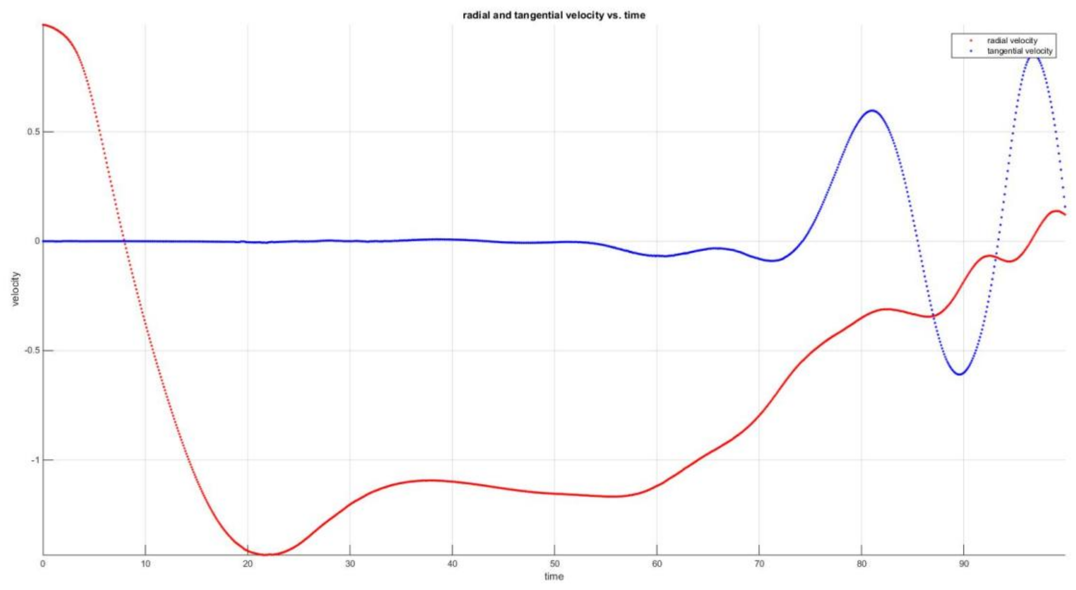
4. Stream function –



5. Velocity magnitude over the cylinder wall



6. **Radial and angular velocity at θ = 0 degree and r = 5.13 m**



Observation –

2-D incompressible Navier-Stokes equation produces vortex shedding as result of viscosity in the fluid. Vortex Shedding is periodic in nature and its frequency depends on geometry of the body, velocity and Reynolds number. A non dimensional number which relates these properties and parameters is Strouhal number.

Strouhal number (St) in vortex shedding over cylindrical bodies is defined as –

$$St = \frac{f_s d}{V_\infty}$$

Where, f_s is vortex shedding frequency, d is diameter of cylinder and V_∞ is free stream velocity.
Here, for *Re = 200* we have St ~ 0.19, *d = 1 m* and V_∞ *= 1 m/s* we have f_s *= 0.19 Hz.* From the radial and tangential velocity graph we have the average crest to crest time is 9.2 sec. The frequency for vortex shading corresponds to half of this time period i.e. *0.21 Hz.* Both the values are in good agreement.

Discussion –

1. Convergence rate of residue largely depends upon the initial guess for the iterative method. Here, if we make initial guess as follows -

$$\psi(i,j) \;= Vinf\ y\left(R(i) - \frac{R(n)^2}{R(i)}\right)\sin(\theta) + 20$$

The convergence is observed to be much faster.

2. The Gauss Seidel method uses values calculated at previous grid points for calculating values at the current grid point in the current iteration cycle. This is the reason for its high convergence rate compared to other methods.
3. The convergence rate for relaxation methods depends upon the relaxation factor, for given discretisation there is an optimal value of relaxation factor for method.
4. Under relaxation makes both the methods slow whereas effect is completely opposite for over relaxation.
5. For higher order scheme the accuracy of the solution is found to be increased. When using a higher order scheme, the points near the boundary need to be discretised using a lower order scheme as you would be forced to take a smaller stencil.
6. Better boundary conditions will provide better result and improve the quality of the solution.

Conclusion –

Vortex shedding is observed from the solution which is immediate consequence of the flow vorticity. The computation time required is very large and can be reduced by using better iterative solutions. Vortex frequency can be calculated more accurately by allowing sufficient time.

Acknowledgement –

I greatly thank you to Dr. Manoj Nair for giving us this assignment and helping us to get a deep insight into the subject. I also appreciate the help provided by Mr. Arun Govind Neelan throughout the work.

Appendix –

MATLAB code – 2-D incompressible Navier Stokes equation

```
clear all; clc;close all

%% Initial conditions

Vinf = 1;

D1 = 1;

Din = D1/2;

D2 = 20*D1;

Dout = D2/2;
%%

m = 40;                                              % length of theta

n = 40;                                             % length of R

omega = 1.93;                                        % relaxation factor

nu = 1/200;                                         % 1/Re

theta = linspace(0,2*pi,m);                         % divisons in theta

R = linspace(Din,Dout,n);                           % divisons in R

dR = R(4) - R(3);                                   % step in R

dt = theta(4) - theta(3);                          % step in theta

t  = linspace(0,100,1000);                          % divison in t

dT = t(4) - t(3);                                  % step in t

w = zeros(n,m);                                    % initializing w = vorticity

xi = zeros(n,m);                                   % initializing streamfunction

%% Getting x and y co-ordinates and boundary conditions of xi.

% getting x and y co-ordinates

for j = 1:m

    for i = 1:n
```

```
        x(i,j) = R(i)*cos(theta(j));

        y(i,j) = R(i)*sin(theta(j));

    end

end

 xi(1,:) = 20;


   for j = 1:m

       xi(n,j) = Vinf*y(n,j) + 20;

   end

%% Main program

W = w;

XI = xi;

for k = 1:length(t) - 1

total_residue = 10;

% Solving poissons equation

while abs(total_residue)>10^(-2)


    % for interior points excluding periodic boundary

for j = 2:m - 1

    for i = 2:n - 1


        a(i,j) = (R(i+1) + R(i))/(2*dR^2);

        b(i,j) = (R(i-1) + R(i))/(2*dR^2);

        c(i,j) = 1/(R(i)*dt^2);

        d(i,j) = 1/(R(i)*dt^2);

        e(i,j) = (2*R(i) + R(i+1) + R(i-1))/(2*dR^2) + 2/(R(i)*dt^2);

        XI(i,j) = (1-omega)*xi(i,j) + (omega/e(i,j))*(a(i,j)*xi(i+1,j) + b(i,j)*XI(i-1,j)
+...
            c(i,j)*xi(i,j+1) + d(i,j)*XI(i,j-1) + R(i)*w(i,j));


    end
```

```
end

% periodic boundary condition

for i = 2:n-1

        a(i,1) = (R(i+1) + R(i))/(2*dR^2);

        b(i,1) = (R(i-1) + R(i))/(2*dR^2);

        c(i,1) = 1/(R(i)*dt^2);

        d(i,1) = 1/(R(i)*dt^2);

        e(i,1) = (2*R(i) + R(i+1) + R(i-1))/(2*dR^2) + 2/(R(i)*dt^2);

        XI(i,1) = (1-omega)*xi(i,1)+ (omega/e(i,1))*(a(i,1)*xi(i+1,1) + b(i,1)*XI(i-1,1)
+...
            c(i,1)*xi(i,1+1) + d(i,1)*XI(i,m-1) + R(i)*w(i,1));

end

XI(:,m) = XI(:,1);

% calculating residue

 total_residue = sqrt(sum(sum((XI - xi).^2)));

  xi = XI;

end

% Extracting velocity from numerical points

for j = 2:m-1

    for i = 1:n

     U_r(i,j) = (1/R(i))*((XI(i,j+1) - XI(i,j-1))/(2*dt));

    end

end

for i = 1:n

    U_r(i,1) = (1/R(i))*((XI(i,2) - XI(i,1))/(dt));

    U_r(i,m) = (1/R(i))*((XI(i,m) - XI(i,m-1))/(dt));

end


for j = 1:m

    for i = 2:n-1
```

```
        U_t(i,j) = -(XI(i+1,j) - XI(i-1,j))/(2*dR);

    end

end


for j = 1:m

    U_t(1,j) = -(XI(2,j) - XI(1,j))/(dR);

    U_t(n,j) = -(XI(n,j) - XI(n-1,j))/(dR);

end

for j = 1:m

    for i = 1:n

        U_x(i,j) = -sin(theta(j))*U_t(i,j) + cos(theta(j))*U_r(i,j);

        U_y(i,j) = sin(theta(j))*U_r(i,j) + cos(theta(j))*U_t(i,j);

        U(i,j) =  sqrt(U_x(i,j)^2 + U_y(i,j)^2);


    end

end


% Vorticity calculations

% at interior points excluding periodic boundary

for j = 2:m - 1

    for i = 2:n - 1

       a1(i,j) = ((-1/dT) + (abs(U_r(i,j))/dR) + abs(U_t(i,j)/R(i))/dt +...
            nu/dR^2 + nu/((R(i)^2)*dt^2));

       b1(i,j) = ( -U_r(i,j)/(2*dR) - abs(U_r(i,j))/(2*dR) - nu/dR^2 +...
            nu/(2*R(i)*dR));

       c1(i,j) = ( - U_t(i,j)/(2*R(i)*dt) - abs(U_t(i,j)/R(i))/(2*dt) -...
            nu/((R(i)^2)*dt^2));

       d1(i,j) = (U_r(i,j)/(2*dR)  - abs(U_r(i,j))/(2*dR) - nu/dR^2 -...
           nu/(2*R(i)*dR));

       e1(i,j) = (U_t(i,j)/(R(i)*2*dt)  - abs(U_t(i,j)/R(i))/(2*dt) -...
            (nu/(R(i)^2*dt^2)));
        W(i,j)  = -dT*(a1(i,j)*w(i,j) + b1(i,j)*w(i-1,j) + c1(i,j)*w(i,j-1) +...
            d1(i,j)*w(i+1,j) + e1(i,j)*w(i,j+1));



    end

end

% periodic boundary condition

for i = 2:n-1



       a1(i,1) = (-1/dT) + (abs(U_r(i,1))/dR) + abs(U_t(i,1))/(R(i)*dt) +...
            nu/dR^2 + nu/(R(i)^2*dt^2);


        b1(i,1) =  -U_r(i,1)/(2*dR) - abs(U_r(i,1))/(2*dR) - nu/dR^2 +...
            nu/(2*R(i)*dR);


        c1(i,1) =  -U_t(i,1)/(2*R(i)*dt) - abs(U_t(i,1))/(2*R(i)*dt) -...
            nu/((R(i)^2)*dt^2);


        d1(i,1) = U_r(i,1)/(2*dR)  - abs(U_r(i,1))/(2*dR) - nu/dR^2 -...
           nu/(2*R(i)*dR);


       e1(i,1) = U_t(i,1)/(R(i)*2*dt)  - abs(U_t(i,1))/(2*R(i)*dt)  -...
            (nu/(R(i)^2*dt^2));


        W(i,1)  = -dT*(a1(i,1)*w(i,1) + b1(i,1)*w(i-1,1) + c1(i,1)*w(i,m-1) +...
            d1(i,1)*w(i+1,1) + e1(i,1)*w(i,1+1));

end

W(:,m) = W(:,1);


W(1,:)  = 2*(((XI(1,:) - XI(2,:))))/dR^2;

W(n,:) = 0;

w = W;

hold on

plot(t(k), U_r(20,1),'.r',t(k), U_t(20,1),'.b')

grid on

axis([0 100 0 20])
```

```
axis tight

legend('radial velocity','tangential velocity')

xlabel('time')

ylabel('velocity')

title('radial and tangential velocity vs. time')

% change following lines to program lines and make above lines comment lines
% for plotting flow over cylinder.

%plot(theta,U(1,:),'-o')

%axis tight

%grid on

%title('velocity over the cylinder')

%xlabel('theta in radians')

%ylabel('velocity')

drawnow;

end


%% Extracting velocity again

for j = 2:m-1

    for i = 1:n

     U_r(i,j) = (1/R(i))*((XI(i,j+1) - XI(i,j-1))/(2*dt));

    end

end

for i = 1:n

    U_r(i,1) = (1/R(i))*((XI(i,2) - XI(i,1))/(dt));

    U_r(i,m) = (1/R(i))*((XI(i,m) - XI(i,m-1))/(dt));

end


for j = 1:m

    for i = 2:n-1


        U_t(i,j) = -(XI(i+1,j) - XI(i-1,j))/(2*dR);
    end

end


for j = 1:m

    U_t(1,j) = -(XI(2,j) - XI(1,j))/(dR);

    U_t(n,j) = -(XI(n,j) - XI(n-1,j))/(dR);

end



%%  getting x and Y component  velocities
for j = 1:m

    for i = 1:n

        U_x(i,j) = -sin(theta(j))*U_t(i,j) + cos(theta(j))*U_r(i,j);

        U_y(i,j) = sin(theta(j))*U_r(i,j) + cos(theta(j))*U_t(i,j);

        U(i,j) =  sqrt(U_x(i,j)^2 + U_y(i,j)^2);


    end

end

figure;

contour(x,y,W,5000);

shading interp;

colorbar

colormap(jet(15));

hold on

color=[0.2 0 0.9]; % r g b color

fill(Din*cos(linspace(0,2*pi)),Din*sin(linspace(0,2*pi)),color);

hold off

axis equal

axis tight

grid on

drawnow;
```

```
title('vorticity')

figure;
quiver(x,y,U_x,U_y);
axis equal
grid on
axis tight
drawnow
title('velocity vector')

figure;
contour(x,y,XI,5000);
shading interp;
colorbar
colormap(jet(15));
hold on
color=[0.2 0 0.9]; % r g b color
fill(Din*cos(linspace(0,2*pi)),Din*sin(linspace(0,2*pi)),color);
hold off
axis equal
axis tight
grid on
drawnow;
title('streamline')


figure;
contour(x,y,U,5000);
shading interp;
colorbar
colormap(jet(15));
```

```
hold on

color=[0.2 0 0.9]; % r g b color

fill(Din*cos(linspace(0,2*pi)),Din*sin(linspace(0,2*pi)),color);

hold off

axis equal

axis tight

grid on

drawnow;

title('velocity magnitude')
```