Python CFD Code for Flow over A Cylinder Block

```
import numpy as np
from matplotlib import pyplot
import numpy.ma as ma

# Inputs 1
m=211;
n=211;
l=2.0;
c=1;
h=1.0;
dx=l/(n-1);
dy=h/(m-1);
x=np.linspace(-5,5,n);
y=np.linspace(-5,5,m);
#dx=np.absolute(x[0]-x[1]);
#dy=np.absolute(y[0]-y[1]);
X,Y=np.meshgrid(x,y);
Z=X+1j*Y;
J=Z;
z=-3.0+1j*0.0;
Z=ma.where(np.absolute(Z-z)<=1.0,1,0);
Z=ma.masked_where(np.absolute(Z-z)<1.0,Z);

# Inputs 2
re=100;
dt=0.001;
velocity=1.0;
```

```
alpha=0.0;
alphar=(alpha*np.pi)/180.0;
u_velocity=velocity*np.cos(alphar)
v_velocity=velocity*np.sin(alphar);
delta=4.5;

##########################################################################################
################################

# Generation of Matrix & Reference Matrix
u_ref=np.zeros((m+1,n),dtype=np.float64);
v_ref=np.zeros((m,n+1),dtype=np.float64);
p_ref=np.zeros((m+1,n+1),dtype=np.float64);
u_matrix=np.zeros((m+1,n),dtype=np.float64);
v_matrix=np.zeros((m,n+1),dtype=np.float64);
p_matrix=np.ones((m+1,n+1),dtype=np.float64);
for i in range(1,m+1):
   for j in range(0,n):
      if not(i==m):
         u_ref[i,j]=Z[i-1,j];
for j in range(1,n+1):
   for i in range(0,m):
      if not(j==n):
         v_ref[i,j]=Z[i,j-1];
         p_ref[i,j]=u_ref[i,j-1];
# Updating' Matrix
u_n=u_matrix;
v_n=v_matrix;
p_n=p_matrix;
```

```
# Colocated Grid

u=np.zeros((m,n),dtype=np.float64);

v=np.zeros((m,n),dtype=np.float64);

p=np.zeros((m,n),dtype=np.float64);

#########################################################################################
#################################

# Boundary Conditions U

u_matrix[1:-1,0]=u_velocity; # Left inlet

u_matrix[0,0:]=-u_matrix[1,0:]; # Top Wall

u_matrix[1:-1,-1]=u_matrix[1:-1,-2];# Right Outlet

u_matrix[-1,0:]=-u_matrix[-2,0:]; # Bottom Wall

for i in range(1,m):

   for j in range(1,n-1):

      if(u_ref[i,j]==1)and(u_ref[i,j+1]==0)and(u_ref[i+1,j]==1)and(u_ref[i,j-1]==0)and(u_ref[i-
1,j]==0):#10100

         u_matrix[i,j]=0.0;

         u_matrix[i,j]=-u_matrix[i-1,j];

      if(u_ref[i,j]==1)and(u_ref[i,j+1]==1)and(u_ref[i+1,j]==1)and(u_ref[i,j-1]==1)and(u_ref[i-
1,j]==0):#11110

         u_matrix[i,j]=-u_matrix[i-1,j];

      if(u_ref[i,j]==1)and(u_ref[i,j+1]==0)and(u_ref[i+1,j]==1)and(u_ref[i,j-1]==1)and(u_ref[i-
1,j]==0):#10110

         u_matrix[i,j]=0.0;

         u_matrix[i,j]=-u_matrix[i-1,j];

      if(u_ref[i,j]==1)and(u_ref[i,j+1]==0)and(u_ref[i+1,j]==1)and(u_ref[i,j-1]==1)and(u_ref[i-
1,j]==1):#10111

         u_matrix[i,j]=0.0;

      if(u_ref[i,j]==1)and(u_ref[i,j+1]==0)and(u_ref[i+1,j]==0)and(u_ref[i,j-1]==1)and(u_ref[i-
1,j]==0):#10010
```

```
        u_matrix[i,j]=0.0;

    if(u_ref[i,j]==1)and(u_ref[i,j+1]==0)and(u_ref[i+1,j]==0)and(u_ref[i,j-1]==1)and(u_ref[i-
1,j]==1):#10011

        u_matrix[i,j]=0.0;

        u_matrix[i,j]=-u_matrix[i+1,j];

    if(u_ref[i,j]==1)and(u_ref[i,j+1]==1)and(u_ref[i+1,j]==0)and(u_ref[i,j-1]==1)and(u_ref[i-
1,j]==1):#11011

        u_matrix[i,j]=u_matrix[i+1,j];

    if(u_ref[i,j]==1)and(u_ref[i,j+1]==0)and(u_ref[i+1,j]==0)and(u_ref[i,j-1]==0)and(u_ref[i-
1,j]==1):#10001

        u_matrix[i,j]=0.0;

        u_matrix[i,j]=-u_matrix[i+1,j];

    if(u_ref[i,j]==1)and(u_ref[i,j+1]==1)and(u_ref[i+1,j]==0)and(u_ref[i,j-1]==0)and(u_ref[i-
1,j]==1):#11001

        u_matrix[i,j]=0.0;

        u_matrix[i,j]=-u_matrix[i+1,j];

    if(u_ref[i,j]==1)and(u_ref[i,j+1]==1)and(u_ref[i+1,j]==1)and(u_ref[i,j-1]==0)and(u_ref[i-
1,j]==1):#11101

        u_matrix[i,j]=0.0;

    if(u_ref[i,j]==1)and(u_ref[i,j+1]==1)and(u_ref[i+1,j]==0)and(u_ref[i,j-1]==0)and(u_ref[i-
1,j]==0):#11000

        u_matrix[i,j]=0.0;

    if(u_ref[i,j]==1)and(u_ref[i,j+1]==1)and(u_ref[i+1,j]==1)and(u_ref[i,j-1]==0)and(u_ref[i-
1,j]==0):#11100

        u_matrix[i,j]=0.0;

        u_matrix[i,j]=u_matrix[i-1,j];


# Boundary Condition V

v_matrix[0:,0]=2*v_velocity-v_matrix[0:,1];# Left Inlet

v_matrix[0,1:-1]=0.0;# Top Wall

v_matrix[0:,-1]=-v_matrix[0:,-2];# Right Outlet
```

```
v_matrix[-1,0:]=0.0; # Bottom Wall

for i in range(1,m):

   for j in range(1,n-1):

      if(v_ref[i,j]==1)and(v_ref[i,j+1]==0)and(v_ref[i+1,j]==1)and(v_ref[i,j-1]==0)and(v_ref[i-
1,j]==0):#10100

         v_matrix[i,j]=0.0;

      if(v_ref[i,j]==1)and(v_ref[i,j+1]==1)and(v_ref[i+1,j]==1)and(v_ref[i,j-1]==1)and(v_ref[i-
1,j]==0):#11110

         v_matrix[i,j]=0.0;

      if(v_ref[i,j]==1)and(v_ref[i,j+1]==0)and(v_ref[i+1,j]==1)and(v_ref[i,j-1]==1)and(v_ref[i-
1,j]==0):#10110

         v_matrix[i,j]=0.0;

         v_n[i,j]=-v_matrix[i,j+1];

      if(v_ref[i,j]==1)and(v_ref[i,j+1]==0)and(v_ref[i+1,j]==1)and(v_ref[i,j-1]==1)and(v_ref[i-
1,j]==1):#10111

         v_matrix[i,j]=-v_matrix[i,j+1];

      if(v_ref[i,j]==1)and(v_ref[i,j+1]==0)and(v_ref[i+1,j]==0)and(v_ref[i,j-1]==1)and(v_ref[i-
1,j]==0):#10010

         v_matrix[i,j]=0.0;

         v_matrix[i,j]=-v_matrix[i,j+1];

      if(v_ref[i,j]==1)and(v_ref[i,j+1]==0)and(v_ref[i+1,j]==0)and(v_ref[i,j-1]==1)and(v_ref[i-
1,j]==1):#10011

         v_matrix[i,j]=0.0;

      if(v_ref[i,j]==1)and(v_ref[i,j+1]==1)and(v_ref[i+1,j]==0)and(v_ref[i,j-1]==1)and(v_ref[i-
1,j]==1):#11011

         v_matrix[i,j]=0.0;

      if(v_ref[i,j]==1)and(v_ref[i,j+1]==0)and(v_ref[i+1,j]==0)and(v_ref[i,j-1]==0)and(v_ref[i-
1,j]==1):#10001

         v_matrix[i,j]=0.0;

      if(v_ref[i,j]==1)and(v_ref[i,j+1]==1)and(v_ref[i+1,j]==0)and(v_ref[i,j-1]==0)and(v_ref[i-
1,j]==1):#11001

         v_matrix[i,j]=0.0;
```

```
        v_matrix[i,j]=-v_matrix[i,j-1];

    if(v_ref[i,j]==1)and(v_ref[i,j+1]==1)and(v_ref[i+1,j]==1)and(v_ref[i,j-1]==0)and(v_ref[i-
1,j]==1):#11101

        v_matrix[i,j]=-v_matrix[i,j-1];

    if(v_ref[i,j]==1)and(v_ref[i,j+1]==1)and(v_ref[i+1,j]==0)and(v_ref[i,j-1]==0)and(v_ref[i-
1,j]==0):#11000

        v_matrix[i,j]=0.0;

        v_matrix[i,j]=-v_matrix[i,j-1];

    if(v_ref[i,j]==1)and(v_ref[i,j+1]==1)and(v_ref[i+1,j]==1)and(v_ref[i,j-1]==0)and(v_ref[i-
1,j]==0):#11100

        v_matrix[i,j]=0.0;

        v_matrix[i,j]=-v_matrix[i,j-1];


# Loop Data

iterations=0;

error_req=1e-1;

error_max=1.0;

error=np.zeros_like(p_matrix);

while iterations<=10:

##########################################################################################
##########################################################################################
####################################

  # X-Momentum Equation

  for i in range(1,m):

    for j in range(1,n-1):

      if not(u_ref[i,j]==1):

        diffusion=((u_matrix[i,j+1]-2*u_matrix[i,j]+u_matrix[i,j-1])/dx**2) +((u_matrix[i+1,j]-
2*u_matrix[i,j]+u_matrix[i-1,j])/dy**2);

        convection1= (((u_matrix[i,j+1]+u_matrix[i,j])/2)**2 -((u_matrix[i,j]+u_matrix[i,j-1])/2)**2)/dx;
```

```
            convection2= ((0.5*(u_matrix[i,j]+u_matrix[i-1,j])*0.5*(v_matrix[i-1,j]+v_matrix[i-1,j+1])) -
(0.5*(u_matrix[i+1,j]+u_matrix[i,j])*0.5*(v_matrix[i,j]+v_matrix[i,j+1])))/dy;

            pressurex= - (p_matrix[i,j+1]-p_matrix[i,j])/dx;

            u_n[i,j]= u_matrix[i,j] -dt*(convection1+convection2)+(dt/re)*diffusion;

   # The Boundary Condition

   for i in range(1,m):

      for j in range(1,n-1):

         if(u_ref[i,j]==1)and(u_ref[i,j+1]==0)and(u_ref[i+1,j]==1)and(u_ref[i,j-1]==0)and(u_ref[i-
1,j]==0):#10100

            u_n[i,j]=0.0;

            u_n[i,j]=-u_n[i-1,j];

         if(u_ref[i,j]==1)and(u_ref[i,j+1]==1)and(u_ref[i+1,j]==1)and(u_ref[i,j-1]==1)and(u_ref[i-
1,j]==0):#11110

            u_n[i,j]=-u_n[i-1,j];

         if(u_ref[i,j]==1)and(u_ref[i,j+1]==0)and(u_ref[i+1,j]==1)and(u_ref[i,j-1]==1)and(u_ref[i-
1,j]==0):#10110

            u_n[i,j]=0.0;

            u_n[i,j]=-u_n[i-1,j];

         if(u_ref[i,j]==1)and(u_ref[i,j+1]==0)and(u_ref[i+1,j]==1)and(u_ref[i,j-1]==1)and(u_ref[i-
1,j]==1):#10111

            u_n[i,j]=0.0;

         if(u_ref[i,j]==1)and(u_ref[i,j+1]==0)and(u_ref[i+1,j]==0)and(u_ref[i,j-1]==1)and(u_ref[i-
1,j]==0):#10010

            u_n[i,j]=0.0;

         if(u_ref[i,j]==1)and(u_ref[i,j+1]==0)and(u_ref[i+1,j]==0)and(u_ref[i,j-1]==1)and(u_ref[i-
1,j]==1):#10011

            u_n[i,j]=0.0;

            u_n[i,j]=-u_n[i+1,j];

         if(u_ref[i,j]==1)and(u_ref[i,j+1]==1)and(u_ref[i+1,j]==0)and(u_ref[i,j-1]==1)and(u_ref[i-
1,j]==1):#11011

            u_n[i,j]=u_n[i+1,j];
```

```
        if(u_ref[i,j]==1)and(u_ref[i,j+1]==0)and(u_ref[i+1,j]==0)and(u_ref[i,j-1]==0)and(u_ref[i-
1,j]==1):#10001
            u_n[i,j]=0.0;
            u_n[i,j]=-u_n[i+1,j];
        if(u_ref[i,j]==1)and(u_ref[i,j+1]==1)and(u_ref[i+1,j]==0)and(u_ref[i,j-1]==0)and(u_ref[i-
1,j]==1):#11001
            u_n[i,j]=0.0;
            u_n[i,j]=-u_n[i+1,j];
        if(u_ref[i,j]==1)and(u_ref[i,j+1]==1)and(u_ref[i+1,j]==1)and(u_ref[i,j-1]==0)and(u_ref[i-
1,j]==1):#11101
            u_n[i,j]=0.0;
        if(u_ref[i,j]==1)and(u_ref[i,j+1]==1)and(u_ref[i+1,j]==0)and(u_ref[i,j-1]==0)and(u_ref[i-
1,j]==0):#11000
            u_n[i,j]=0.0;
        if(u_ref[i,j]==1)and(u_ref[i,j+1]==1)and(u_ref[i+1,j]==1)and(u_ref[i,j-1]==0)and(u_ref[i-
1,j]==0):#11100
            u_n[i,j]=0.0;
            u_n[i,j]=u_n[i-1,j];


   u_n[1:-1,0]=u_velocity; # Left inlet
   u_n[0,0:]=-u_n[1,0:]; # Top Wall
   u_n[1:-1,-1]=u_n[1:-1,-2];# Right Outlet
   u_n[-1,0:]=-u_n[-2,0:]; # Bottom Wall
##########################################################################################
##########################################################################################
###################################


   # Y-Momentum Equation
   for i in range(1,m-1):
      for j in range(1,n):
         if not(v_ref[i,j]==1):
```

```
            diffusion=((v_matrix[i,j+1]-2*v_matrix[i,j]+v_matrix[i,j-1])/dx**2) +((v_matrix[i+1,j]-
2*v_matrix[i,j]+v_matrix[i-1,j])/dy**2);

            convection1= ((0.5*(v_matrix[i,j]+v_matrix[i,j+1])*0.5*(u_matrix[i,j]+u_matrix[i+1,j])) -
(0.5*(v_matrix[i,j]+v_matrix[i,j-1])*0.5*(u_matrix[i,j-1]+u_matrix[i+1,j-1])))/dx;

            convection2= (((v_matrix[i,j]+v_matrix[i-1,j])*0.5)**2 -((v_matrix[i,j]+v_matrix[i+1,j]))**2)/dy;

            pressurey= - (p_matrix[i+1,j]-p_matrix[i,j])/dy;

            v_n[i,j]= v_matrix[i,j]- dt*(convection1+convection2)+(dt/re)*diffusion;


   # The Boundary Condition

   for i in range(1,m):

      for j in range(1,n-1):

         if(v_ref[i,j]==1)and(v_ref[i,j+1]==0)and(v_ref[i+1,j]==1)and(v_ref[i,j-1]==0)and(v_ref[i-
1,j]==0):#10100

            v_n[i,j]=0.0;

         if(v_ref[i,j]==1)and(v_ref[i,j+1]==1)and(v_ref[i+1,j]==1)and(v_ref[i,j-1]==1)and(v_ref[i-
1,j]==0):#11110

            v_n[i,j]=0.0;

         if(v_ref[i,j]==1)and(v_ref[i,j+1]==0)and(v_ref[i+1,j]==1)and(v_ref[i,j-1]==1)and(v_ref[i-
1,j]==0):#10110

            v_n[i,j]=0.0;

            v_n[i,j]=-v_n[i,j+1];

         if(v_ref[i,j]==1)and(v_ref[i,j+1]==0)and(v_ref[i+1,j]==1)and(v_ref[i,j-1]==1)and(v_ref[i-
1,j]==1):#10111

            v_n[i,j]=-v_n[i,j+1];

         if(v_ref[i,j]==1)and(v_ref[i,j+1]==0)and(v_ref[i+1,j]==0)and(v_ref[i,j-1]==1)and(v_ref[i-
1,j]==0):#10010

            v_n[i,j]=0.0;

            v_n[i,j]=-v_n[i,j+1];

         if(v_ref[i,j]==1)and(v_ref[i,j+1]==0)and(v_ref[i+1,j]==0)and(v_ref[i,j-1]==1)and(v_ref[i-
1,j]==1):#10011

            v_n[i,j]=0.0;
```

```
        if(v_ref[i,j]==1)and(v_ref[i,j+1]==1)and(v_ref[i+1,j]==0)and(v_ref[i,j-1]==1)and(v_ref[i-
1,j]==1):#11011

            v_n[i,j]=0.0;

        if(v_ref[i,j]==1)and(v_ref[i,j+1]==0)and(v_ref[i+1,j]==0)and(v_ref[i,j-1]==0)and(v_ref[i-
1,j]==1):#10001

            v_n[i,j]=0.0;

        if(v_ref[i,j]==1)and(v_ref[i,j+1]==1)and(v_ref[i+1,j]==0)and(v_ref[i,j-1]==0)and(v_ref[i-
1,j]==1):#11001

            v_n[i,j]=0.0;

            v_n[i,j]=-v_n[i,j-1];

        if(v_ref[i,j]==1)and(v_ref[i,j+1]==1)and(v_ref[i+1,j]==1)and(v_ref[i,j-1]==0)and(v_ref[i-
1,j]==1):#11101

            v_n[i,j]=-v_n[i,j-1];

        if(v_ref[i,j]==1)and(v_ref[i,j+1]==1)and(v_ref[i+1,j]==0)and(v_ref[i,j-1]==0)and(v_ref[i-
1,j]==0):#11000

            v_n[i,j]=0.0;

            v_n[i,j]=-v_n[i,j-1];

        if(v_ref[i,j]==1)and(v_ref[i,j+1]==1)and(v_ref[i+1,j]==1)and(v_ref[i,j-1]==0)and(v_ref[i-
1,j]==0):#11100

            v_n[i,j]=0.0;

            v_n[i,j]=-v_n[i,j-1];


    v_n[0:,0]=2*v_velocity-v_n[0:,1];# Left Inlet

    v_n[0,1:-1]=0.0;# Top Wall

    v_n[0:,-1]=-v_n[0:,-2];# Right Outlet

    v_n[-1,0:]=0.0; # Bottom Wall

##########################################################################################
##########################################################################################
####################################

    # Continuity Equation

    for i in range(1,m):
```

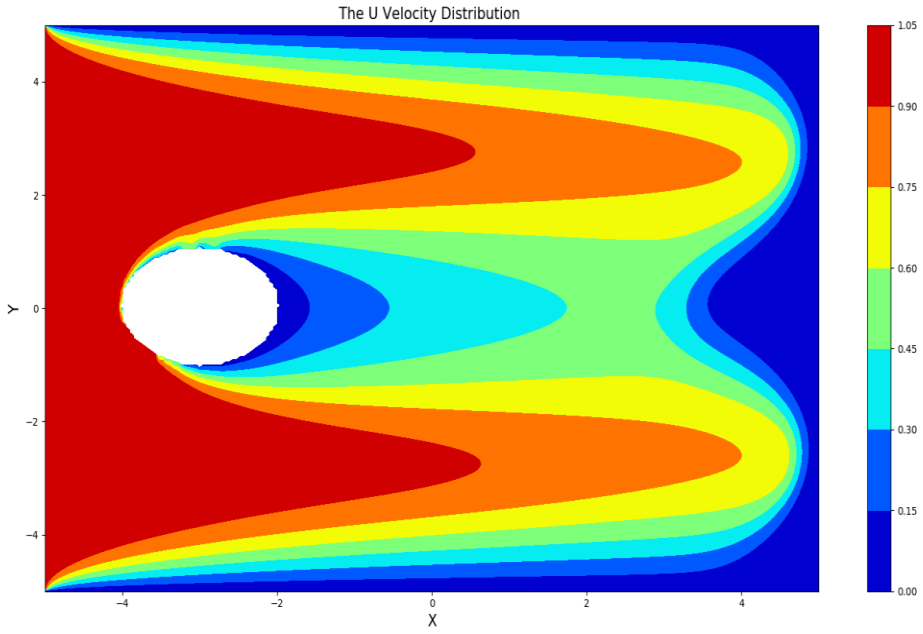
```
    for j in range(1,n):
        if not (p_ref[i,j]==1):
            p_n[i,j]=p_matrix[i,j]- (delta*dt)*((u_matrix[i,j]-u_matrix[i,j-1])/dx + (v_matrix[i-1,j]-
v_matrix[i,j])/dy );


  # Boundary Condition
  for i in range(1,m):
    for j in range(1,n):
        if(p_ref[i,j]==1)and(p_ref[i,j+1]==0)and(p_ref[i+1,j]==1)and(p_ref[i,j-1]==0)and(p_ref[i-
1,j]==0):#10100
            p_n[i,j]=p_n[i-1,j];
        if(p_ref[i,j]==1)and(p_ref[i,j+1]==1)and(p_ref[i+1,j]==1)and(p_ref[i,j-1]==1)and(p_ref[i-
1,j]==0):#11110
            p_n[i,j]=p_n[i-1,j];
        if(p_ref[i,j]==1)and(p_ref[i,j+1]==0)and(p_ref[i+1,j]==1)and(p_ref[i,j-1]==1)and(p_ref[i-
1,j]==0):#10110
            p_n[i,j]=p_n[i,j+1];
            p_n[i,j]=p_n[i-1,j];
        if(p_ref[i,j]==1)and(p_ref[i,j+1]==0)and(p_ref[i+1,j]==1)and(p_ref[i,j-1]==1)and(p_ref[i-
1,j]==1):#10111
            p_n[i,j]=p_n[i,j+1];
        if(p_ref[i,j]==1)and(p_ref[i,j+1]==0)and(p_ref[i+1,j]==0)and(p_ref[i,j-1]==1)and(p_ref[i-
1,j]==0):#10010
            p_n[i,j]=p_n[i,j+1];
        if(p_ref[i,j]==1)and(p_ref[i,j+1]==0)and(p_ref[i+1,j]==0)and(p_ref[i,j-1]==1)and(p_ref[i-
1,j]==1):#10011
            p_n[i,j]=p_n[i,j+1];
            p_n[i,j]=p_n[i+1,j];
        if(p_ref[i,j]==1)and(p_ref[i,j+1]==1)and(p_ref[i+1,j]==0)and(p_ref[i,j-1]==1)and(p_ref[i-
1,j]==1):#11011
            p_n[i,j]=p_n[i+1,j];
```

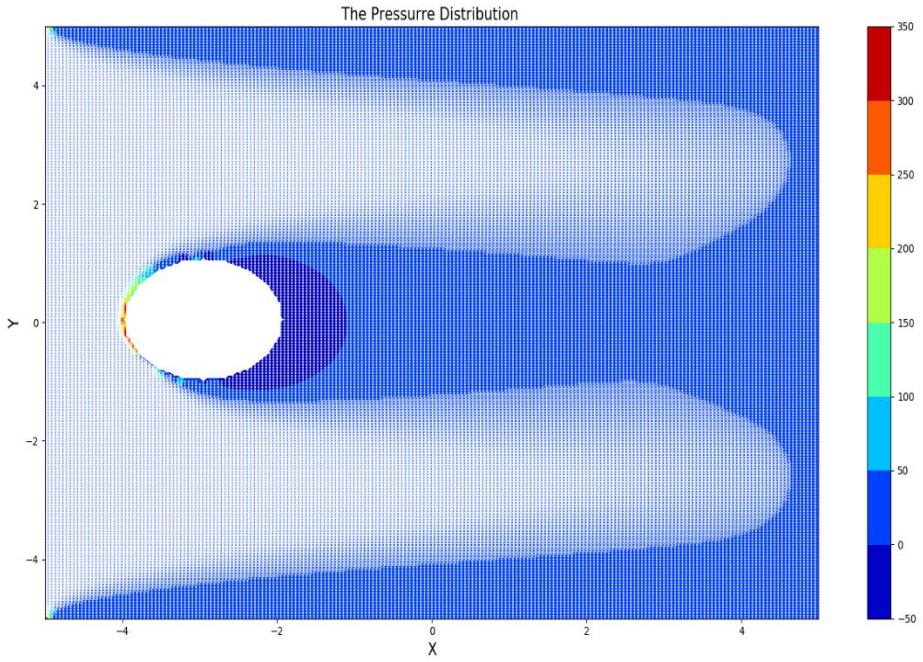
```
        if(p_ref[i,j]==1)and(p_ref[i,j+1]==0)and(p_ref[i+1,j]==0)and(p_ref[i,j-1]==0)and(p_ref[i-
1,j]==1):#10001

            p_n[i,j]=p_n[i+1,j];

        if(p_ref[i,j]==1)and(p_ref[i,j+1]==1)and(p_ref[i+1,j]==0)and(p_ref[i,j-1]==0)and(p_ref[i-
1,j]==1):#11001

            p_n[i,j]=p_n[i,j-1];

            p_n[i,j]=p_n[i+1,j];

        if(p_ref[i,j]==1)and(p_ref[i,j+1]==1)and(p_ref[i+1,j]==1)and(p_ref[i,j-1]==0)and(p_ref[i-
1,j]==1):#11101

            p_n[i,j]=p_n[i,j-1];

        if(p_ref[i,j]==1)and(p_ref[i,j+1]==1)and(p_ref[i+1,j]==0)and(p_ref[i,j-1]==0)and(p_ref[i-
1,j]==0):#11000

            p_n[i,j]=p_n[i,j-1];

        if(p_ref[i,j]==1)and(p_ref[i,j+1]==1)and(p_ref[i+1,j]==1)and(p_ref[i,j-1]==0)and(p_ref[i-
1,j]==0):#11100

            p_n[i,j]=p_n[i,j-1];

            p_n[i,j]=p_n[i-1,j]


   p_n[0:,0]=p_n[0:,1]; # Left inlet

   p_n[-1,0:]=p_n[-2,0:]; # Bottom Wall

   p_n[0:,-1]=p_n[0:,-2]; # Right Outlet

   p_n[0,0:]=p_n[1,0:]; # Top Wall


##########################################################################################
##########################################################################################
####################################

   error=np.zeros_like(p_matrix);

   for i in range(1,m):

      for j in range(1,n):

         if not (p_ref[i,j]==1):

            error[i,j]= error[i,j] +np.abs((u_n[i,j]-u_n[i,j-1])/float(dx)+(v_n[i-1,j]-v_n[i,j])/float(dy));
```

```
    error_max=np.abs(np.max(error));
    print(error_max);

     #Plot the Residual Error
    if(iterations%25==0):
        print(iterations);
#
#     pyplot.semilogy(iterations,error_max,color='r',marker='.');
#     pyplot.xlabel('Iterations',fontsize=12);
#     pyplot.ylabel('Residual Error',fontsize=12);
#     pyplot.title('Residuals Plot',fontsize=12);

    # Update New Velocity and Pressure
    u_matrix=u_n;
    v_matrix=v_n;
    p_matrix=p_n;
    iterations= iterations +1;



# Map into collocated grid
for i in range(0,m):
    for j in range(0,n):
        if not (u_ref[i,j]==1):
            u[i,j]=0.5*(u_matrix[i,j]+u_matrix[i+1,j]);
        if not (v_ref[i,j]==1):
            v[i,j]=0.5*(v_matrix[i,j]+v_matrix[i,j+1]);
        if not (p_ref[i,j]==1):
            p[i,j]=0.25*(p_matrix[i,j]+p_matrix[i,j+1]+p_matrix[i+1,j]+p_matrix[i+1,j+1]);
#a=v[0,0]
```

```
u=ma.masked_where(u==0,u)
p=ma.masked_where(p==0,p)
u[0,0:]=0;
u[-1,0:]=0;


pyplot.figure()
pyplot.contourf(X,Y,u,cmap='jet');
pyplot.colorbar();
#pyplot.quiver(X,Y,u,v,color='w',headlength=1.5,headwidth=2,linewidth=0.1);
pyplot.xlabel('X',fontsize=15);
pyplot.ylabel('Y',fontsize=15);
pyplot.title('The U Velocity Distribution ',fontsize=15);


pyplot.figure(edgecolor='k');
pyplot.contourf(X,Y,p,cmap='jet',origin='lower');
pyplot.colorbar();
#pyplot.quiver(X,Y,u,v,color='w',headlength=1.5,headwidth=2,linewidth=0.1);
pyplot.xlabel('X',fontsize=15);
pyplot.ylabel('Y',fontsize=15);
pyplot.title('The Pressurre Distribution ',fontsize=15);
```

RESULTS


The U Velocity Distribution
Y
X
4
2
0
−2
−4
−4
−2
0
2
4
1.05
0.90
0.75
0.60
0.45
0.30
0.15
0.00


The Pressurre Distribution
Y
X
4
2
0
−2
−4
−4
−2
0
2
4
350
300
250
200
150
100
50
0
−50