Python CFD Code for Flow over A Rectangular Block

```
import numpy as np

from matplotlib import pyplot

import numpy.ma as ma



def get_maskedMatrix(dx,dy,m,n,na,nb,nd,ne,matrix,u_matrix,v_matrix,p_matrix):
    # Matrix masked
    for i in range(0,m):
        for j in range(0,n):
            if ((j>na) and (j<nb) and (i>nd) and (i<ne)):
                matrix[i,j]=2.0;
    matrix=ma.masked_where(matrix==2.0,matrix);

    # Masking U_Matrix
    for i in range(0,m+1):
        for j in range(0,n):
            if ((j>na) and (j<nb) and (i>nd+1) and (i<ne)):
                u_matrix[i,j]=2.0;
    u_matrix=ma.masked_where(u_matrix==2.0,u_matrix);

    # Masking V Matrix
    for i in range(0,m):
        for j in range(0,n+1):
            if ((j>na+1) and (j<nb) and (i>nd) and (i<ne)):
                v_matrix[i,j]=2.0;
    v_matrix=ma.masked_where(v_matrix==2.0,v_matrix);
```

```
    # Masking P Matrix
    for i in range(0,m+1):
        for j in range(0,n+1):
            if ((j>na+1) and (j<nb) and (i>nd+1) and (i<ne)):
                p_matrix[i,j]=2.0;
    p_matrix=ma.masked_where(p_matrix==2.0,p_matrix);
    return matrix,u_matrix,v_matrix,p_matrix;



# Inputs
m=51; # No of points in i direction.
n=51; # No of points in j direction.
l=1.3;
h=0.9;
dx=l/(n-1); # dx
dy=h/(m-1);  # dy
a=0.2*l;b=0.2*l;c=0.6*l; # Percentage of division of length
d=0.4*h;e=0.2*h;f=0.4*h; # Percentage of division of height
na=int(np.ceil(a/dx)); # Starting point of block horizontally
nb=int(np.ceil(b/dx));
nd=int(np.floor(d/dy));# Starting point of block vertically
ne=int(np.floor(e/dy));
nb=nb+na; # Ending point of block horizontally.
ne=nd+ne; # Ending point of block vertically.
dt=0.001;
velocity=2;
re=10;
delta=4.5;
matrix=np.zeros((m,n),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);


x=np.linspace(0,l,n);

y=np.linspace(0,h,m);

X,Y=np.meshgrid(x,y);


matrix,u_matrix,v_matrix,p_matrix=get_maskedMatrix(dx,dy,m,n,na,nb,nd,ne,matrix,u_matrix,v_matrix
,p_matrix);

# Collocated Grid

u=np.zeros_like(matrix);

v=np.zeros_like(matrix);

p=np.zeros_like(matrix);

u[1:-1,0]=velocity;

# Input the boundary conditions

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

u_matrix[1:-1,0]=velocity; # Left Inlet

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

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

u_matrix[nd+1:ne+1,na]=0.0; # Left Cylinder Wall

u_matrix[nd+1:ne+1,nb]=0.0 # Right Cylinder Wall

u_matrix[nd+1,na:nb+1]=-u_matrix[nd,na:nb+1]# Top Cylinder Wall

u_matrix[ne,na:nb+1]=-u_matrix[ne+1,na:nb+1]# Bottom Cylinder Wall



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

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

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

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

v_matrix[nd:ne+1,na+1]=-v_matrix[nd:nd+1,na];# Left Cylinder wall

v_matrix[nd:ne+1,nb]=-v_matrix[nd:ne+1,nb+1]; # Right Cylinder Wall

v_matrix[nd,na+1:nb+1]=0.0; #  Top Cylinder Wall

v_matrix[ne,na+1:nb+1]=0.0 # Bottom Cylinder Wall


# Updating Matrix

un_matrix=u_matrix;

vn_matrix=v_matrix;

pn_matrix=p_matrix;


# Solving

error_max=1.0;

iterations=0;

error_req=1e-1;

fig=pyplot.figure();


while iterations<=100:#error_max>=error_req:

  # Solve X-Momentum Equation

  for i in range(1,m):

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

      if not ((j>=na)and(j<=nb)and(i>nd)and(i<ne+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;

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

```
#    # Apply Boundary Condition for X-Momentum New Variable
    un_matrix[0,0:]=-un_matrix[1,0:] # Top Wall
    un_matrix[1:-1,0]=velocity; # Left Inlet
    un_matrix[-1,0:]=-un_matrix[-2,0:]; # Bottom Wall
    un_matrix[1:-1,-1]=un_matrix[1:-1,-2]; # Right Outlet
    un_matrix[nd+1:ne+1,na]=0.0; # Left Cylinder Wall
    un_matrix[nd+1:ne+1,nb]=0.0 # Right Cylinder Wall
    un_matrix[nd+1,na:nb+1]=-un_matrix[nd,na:nb+1]# Top Cylinder Wall
    un_matrix[ne,na:nb+1]=-un_matrix[ne+1,na:nb+1]# Bottom Cylinder Wall


    for i in range(1,m-1):
        for j in range(1,n):
            if not((j>na)and(j<nb+1)and(i>=nd)and(i<=ne)):
                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;
                vn_matrix[i,j]= v_matrix[i,j]- dt*(convection1+convection2)+(dt/re)*diffusion;
#    # Apply Boundary Condition for Y-Momentum New Variable
    vn_matrix[0,1:-1]=0.0; # Top Wall
    vn_matrix[0:,0]=-vn_matrix[0:,1];# Left Inlet
    vn_matrix[-1,1:-1]=0.0;# Bottom Wall
    vn_matrix[0:,-1]=-vn_matrix[0:,-2]; # Right Outlet
    vn_matrix[nd:ne+1,na+1]=-vn_matrix[nd:nd+1,na];# Left Cylinder wall
    vn_matrix[nd:ne+1,nb]=-vn_matrix[nd:ne+1,nb+1]; # Right Cylinder Wall
    vn_matrix[nd,na+1:nb+1]=0.0; #  Top Cylinder Wall
    vn_matrix[ne,na+1:nb+1]=0.0 # Bottom Cylinder Wall
```

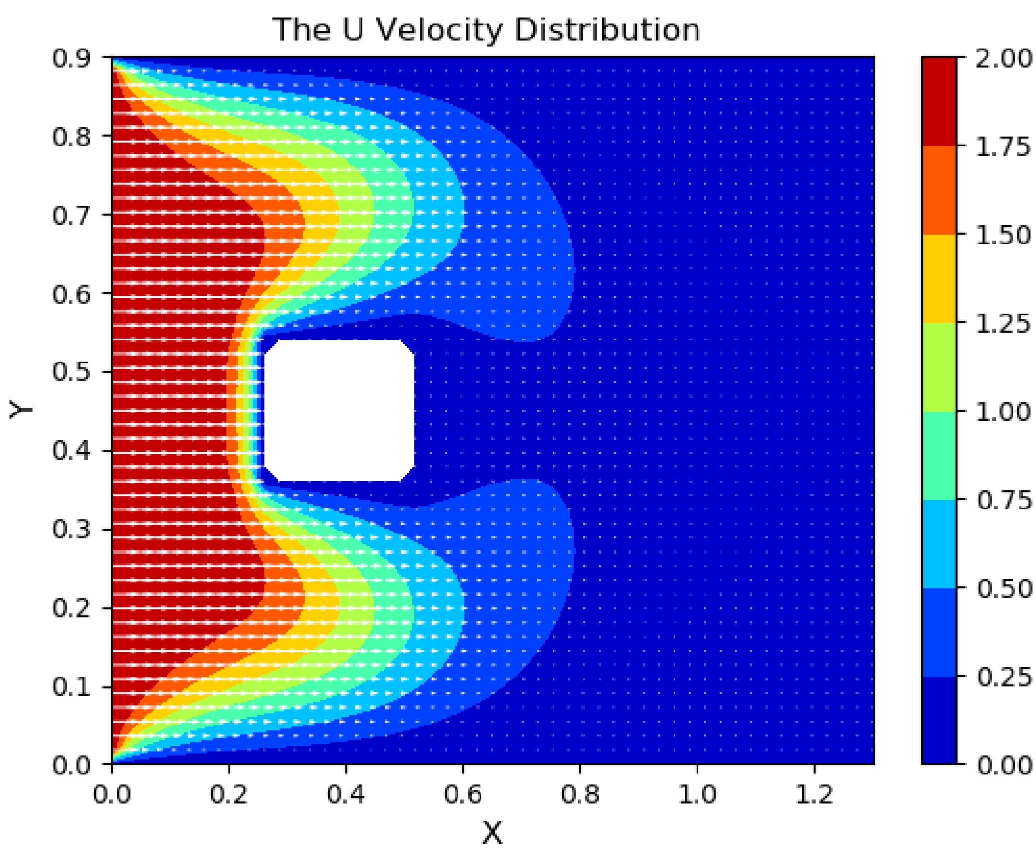
```
    for i in range(1,m):
        for j in range(1,n):
            if not ((j>na) and (j<=nb) and (i>nd) and (i<=ne)):
                pn_matrix[i,j]=p_matrix[i,j]- (delta*dt)*((u_matrix[i,j]-u_matrix[i,j-1])/dx + (v_matrix[i-1,j]-
v_matrix[i,j])/dy );
#    # Apply Pressure Boundary Condition
    pn_matrix[0,0:]=pn_matrix[1,0:];# Top Wall
    pn_matrix[0:,0]=pn_matrix[0:,1];# Left Inlet
    pn_matrix[-1,0:]=pn_matrix[-2,0:];# Bottom Wall
    pn_matrix[0:,-1]=pn_matrix[0:,-2]; # Right Outlet
    pn_matrix[nd+1,na+1:nb+1]=pn_matrix[nd,na+1:nb+1] # Top Wall of block
    pn_matrix[nd+1:ne+1,na+1]=pn_matrix[nd+1:ne+1,na]; # Left Wall of block
    pn_matrix[ne,na+1:nb+1]=pn_matrix[ne+1,na+1:nb+1]; # Bottom Wall of Block
    pn_matrix[nd+1:ne+1,nb]=pn_matrix[nd+1:ne+1,nb+1]; # Right Wall of block
    # Continuity Error Residual
    error=np.zeros_like(p_matrix);
    for i in range(1,m):
        for j in range(1,n):
            if not ((j>na) and (j<=nb) and (i>nd) and (i<=ne)):
                error[i,j]= error[i,j] + np.abs((un_matrix[i,j]-un_matrix[i,j-1])/float(dx)+(vn_matrix[i-1,j]-
vn_matrix[i,j])/float(dy));
    error_max=np.abs(np.max(error));
     #Plot the Residual Error
    if(iterations%10==0):
        print(error_max)
#      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);
```

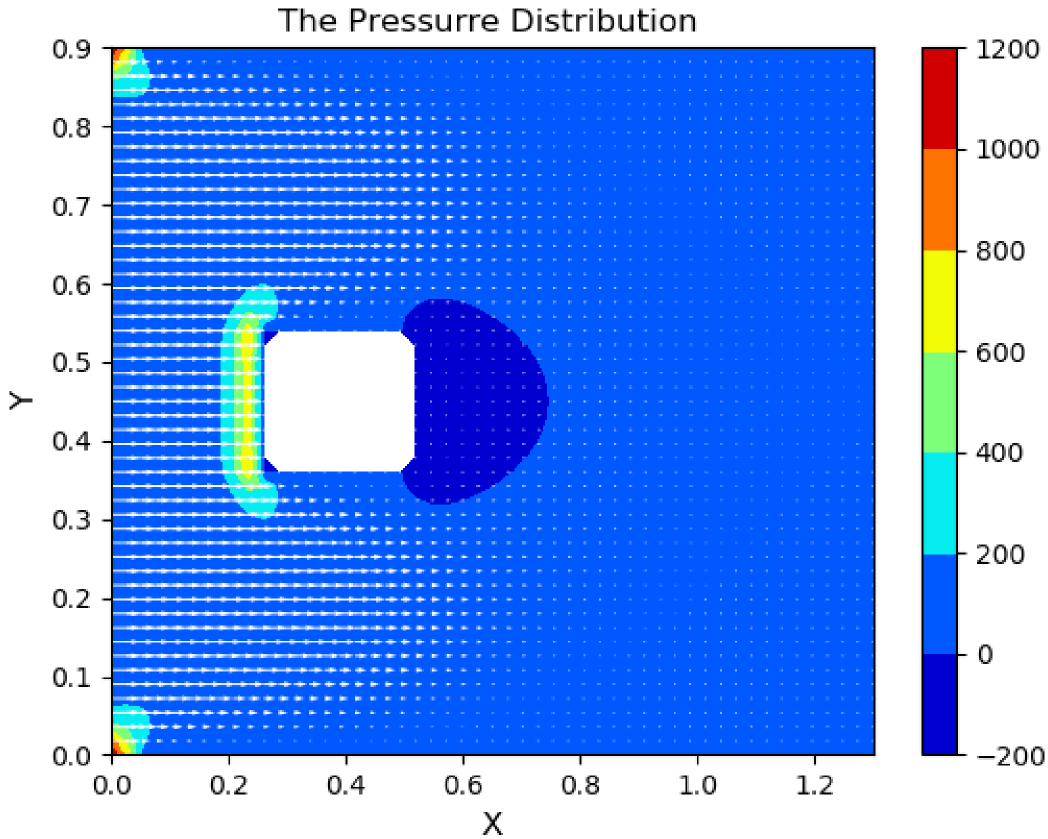
```
    # Update New Velocity and Pressure
    u_matrix=un_matrix;
    v_matrix=vn_matrix;
    p_matrix=pn_matrix;
    iterations= iterations +1;
# Map into collocated grid
for i in range(0,m):
    for j in range(0,n):
        if not ((j>=na)and(j<=nb)and(i>=nd)and(i<=ne)):
            u[i,j]=0.5*(u_matrix[i,j]+u_matrix[i+1,j]);
            v[i,j]=0.5*(v_matrix[i,j]+v_matrix[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]);
#  Plotting
pyplot.figure()
pyplot.contourf(X,Y,u,cmap='jet');
pyplot.colorbar();
pyplot.quiver(X,Y,u,v,color='w',linewidth=0.1);
pyplot.xlabel('X',fontsize=12);
pyplot.ylabel('Y',fontsize=12);
pyplot.title('The U Velocity Distribution ',fontsize=12);

pyplot.figure();
pyplot.contourf(X,Y,v,cmap='jet');
pyplot.colorbar();
pyplot.quiver(X,Y,u,v,color='w',linewidth=0.1);
pyplot.xlabel('X',fontsize=12);
pyplot.ylabel('Y',fontsize=12);
pyplot.title('The V Velocity Distribution ',fontsize=12);
```

```
pyplot.figure(edgecolor='k');
pyplot.contourf(X,Y,p,cmap='jet',origin='lower');
pyplot.colorbar();
pyplot.quiver(X,Y,u,v,color='w',linewidth=0.1);
pyplot.xlabel('X',fontsize=12);
pyplot.ylabel('Y',fontsize=12);
pyplot.title('The Pressurre Distribution ',fontsize=12);
```

RESULTS





The Pressurre Distribution
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Y
0.0
0.2
0.4
0.6
0.8
1.0
1.2
X
1200
1000
800
600
400
200
0
−200