# Assignment 3 Report

Mannem S V Sayi Teja Reddy

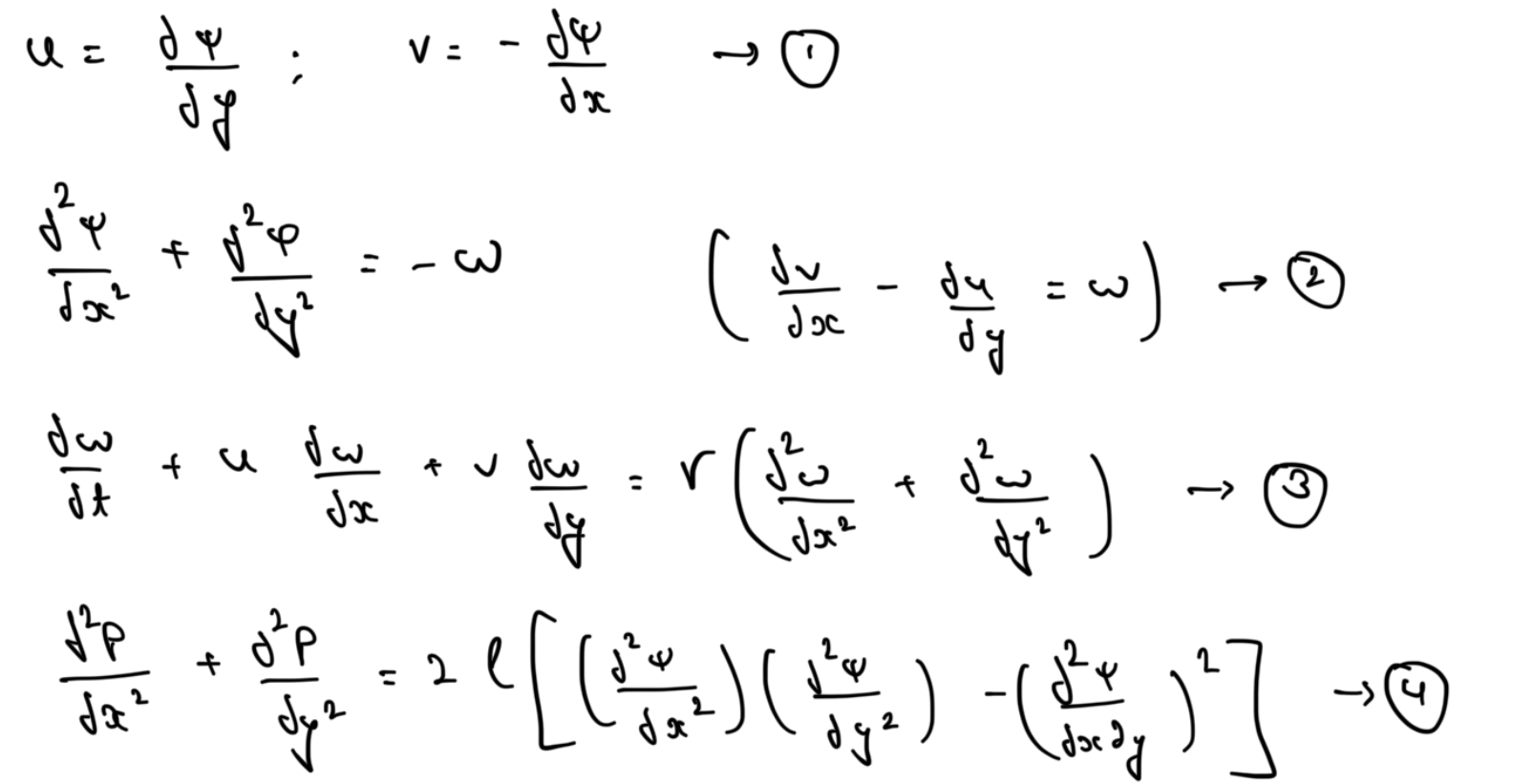
ME20B108

## Problem Definition

**Lid-driven cavity**

Consider a square cavity of side, a=1 unit as shown in Figure 1. The cavity is filled with an incompressible fluid. Assume the flow to be a steady, 2-dimensional flow. Determine the fluid flow pattern and pressure distribution inside the cavity at Reynolds number, Re = 100 using any finite difference method you have learned from this AM5630 course (i.e. Stream function vorticity method, SMAC method). Use the non-dimensional form of the governing equation with appropriate boundary conditions.

## Governing Equation

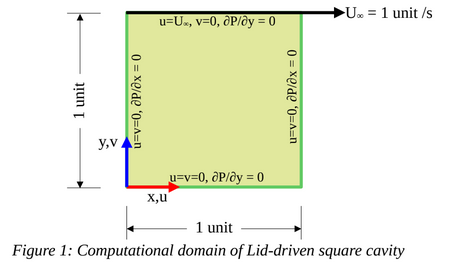


## Initial condition (if any) / Boundary Conditions

Top wall velocity = 1 unit/s

All other walls = 0 unit/s

If other IC and BC required, assume appropriate conditions (Refer Figure 1).



## Numerical Formulation

The stream function-vorticity method is a numerical technique commonly used to solve fluid flow problems, particularly in computational fluid dynamics (CFD). It's based on expressing the fluid flow velocity as a combination of a stream function (ψ) and a vorticity (ω) function.

Here's a brief overview of the numerical formulation of the stream function-vorticity method:

Governing Equations:

The method is typically applied to incompressible, two-dimensional flow governed by the Navier-Stokes equations. The equations for conservation of mass (continuity) and momentum (Navier-Stokes) are simplified and rewritten in terms of stream function and vorticity.

Stream Function (ψ):

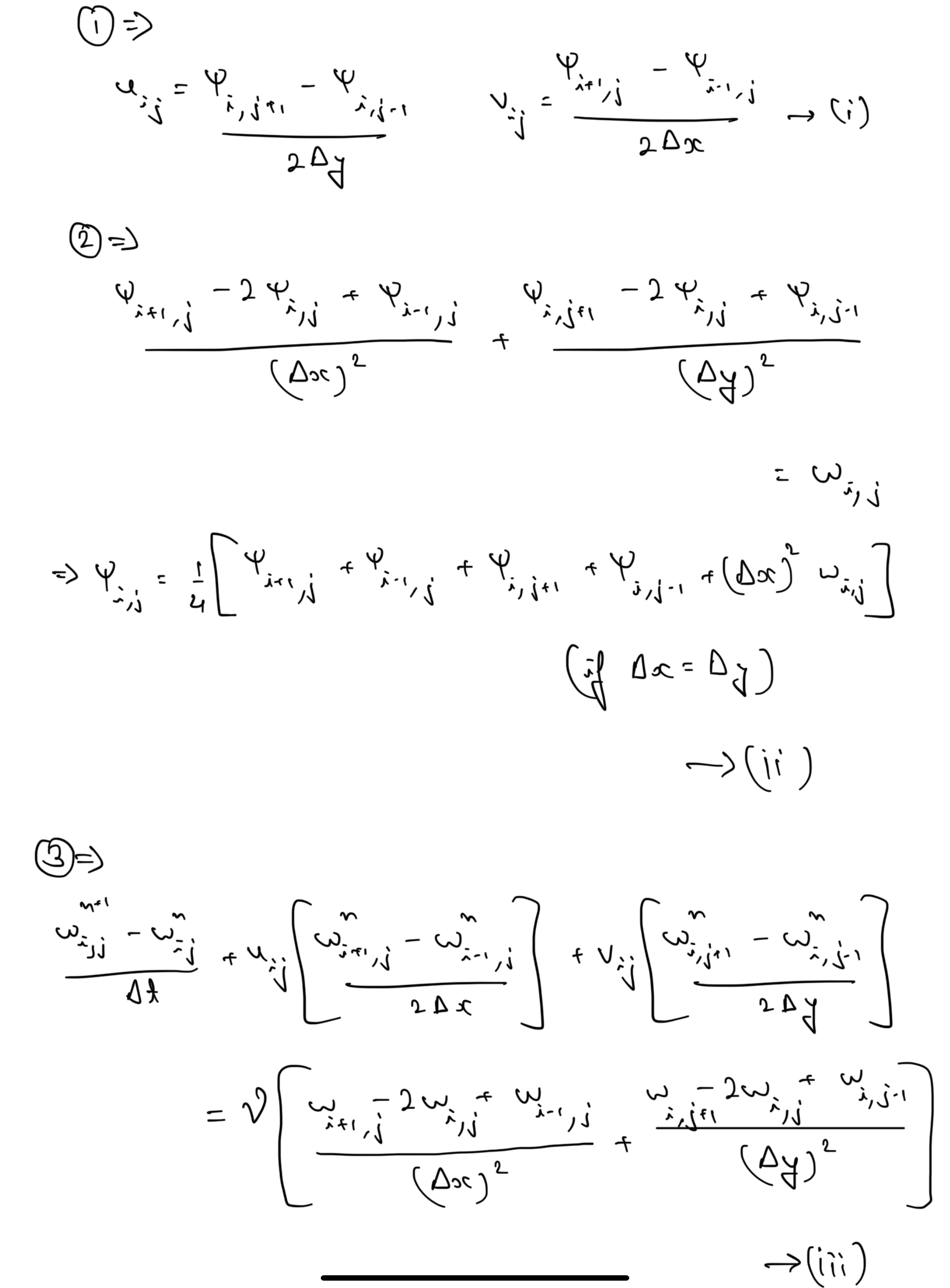
The stream function is a scalar field that describes the flow velocity components in terms of a single scalar function. In two-dimensional, incompressible flow, it satisfies the Laplace equation:

∇2ψ=−ω∇2ψ=−ω

where ∇² is the Laplacian operator and ω is the vorticity.

We use Finite difference method to solve these above equations numerically.

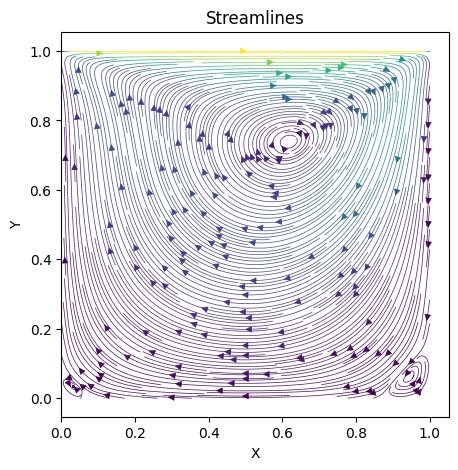
## Flow chart or Pseudo code

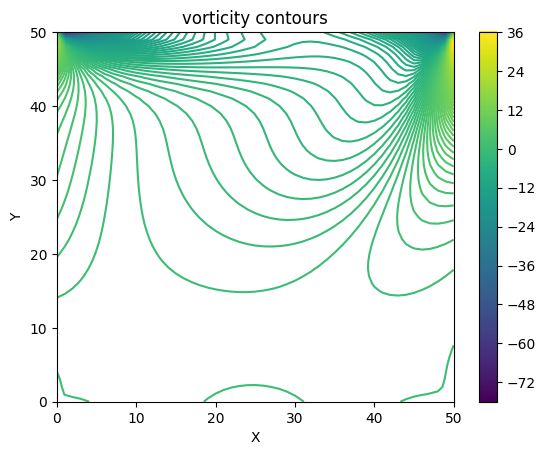


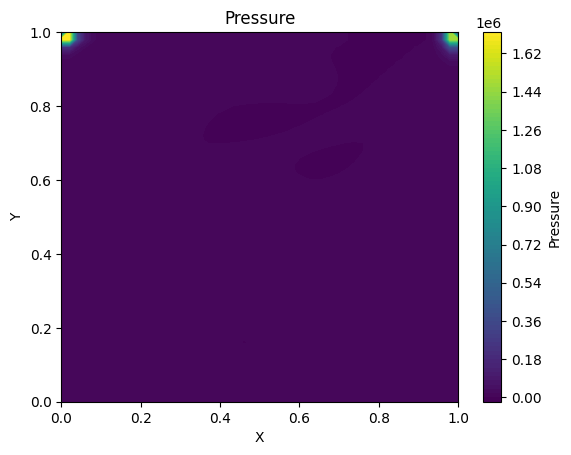
1. Declare all the variables allocate the array, initialize all the dependent variable.
2. Assuming ωi,j = 0, calculate the values of ψi,j using equation (i)
3. Calculate the values of ui,j and vi,j for the equations (ii).
4. Solve equation (iii) to get ωi,j values.
5. Check if defined error if less than the tolerance or not.
6. Go to step 2 and repeat till the tolerance condition is met.

## Results and discussion

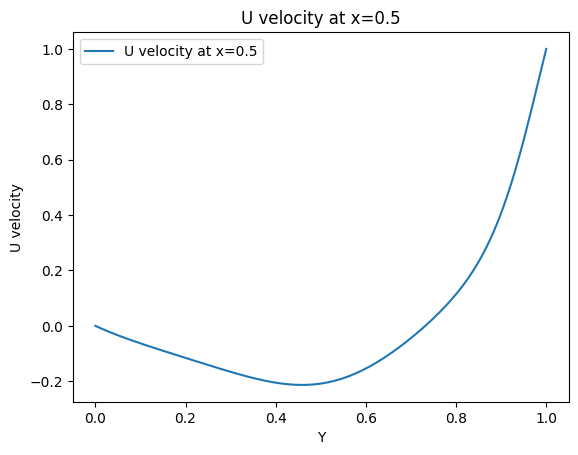
1. Plot stream lines

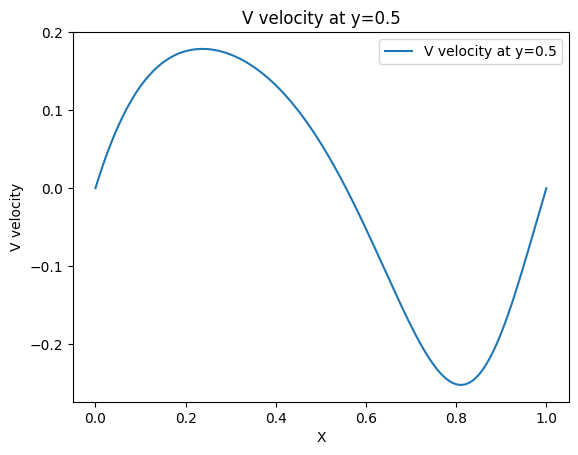


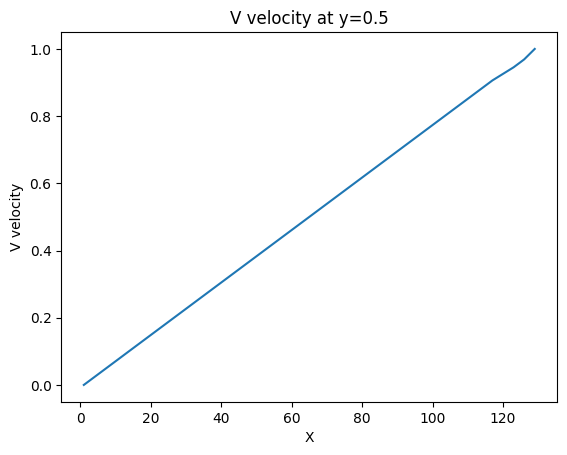
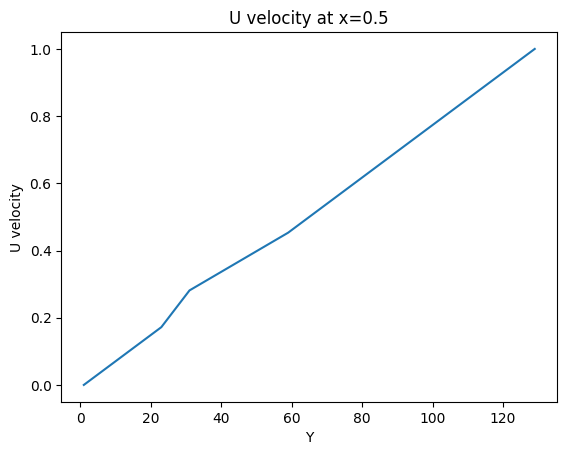
2. Plot vorticity contours

3. Plot the pressure distribution

4. Plot the ‘u’ Vs ‘y’ at x = 0.5 unit



5. Plot the ‘v’ Vs ‘x’ at y = 0.5 unit

6. Compare the x-y plots (from 4 and 5) with Ghia et al. (1982) data.

## Appendix: Computer Code

import numpy as np

import matplotlib.pyplot as plt

nx = 129

ny = 129

lx = 1.0

ly = 1.0

dx = lx/(nx-1)

dy = ly/(ny-1)

dt = 0.001

tol = 1e-6

err = 1.0

Re = 100.0

Ut = 1.0 # u top wall

Ub = 0.0 # u bottom wall

Vl = 0.0 # V left wall

Vr = 0.0 # V right wall

Nue = 1.0/Re

psi = np.zeros([ny,nx])

omega = np.zeros([ny,nx])

u = np.zeros([ny,nx])

v = np.zeros([ny,nx])

u[-1,1:-1] = Ut

# apply boundary conditions

omega[0,1:-1] = -2.0\*psi[1, 1:-1]/dy\*\*2 # bottom wall

omega[-1,1:-1] = -2.0\*psi[-2, 1:-1]/dy\*\*2 - 2.0\*Ut/dy # vorticity on top wall (moving at Uwall)

omega[1:-1,-1] = -2.0\*psi[1:-1,-2]/dx\*\*2 # right wall

omega[1:-1,0] = -2.0\*psi[1:-1,1]/dx/dx # left wall

u[1:-1,1:-1] = (psi[2:,1:-1] - psi[:-2,1:-1])/2.0/dy

v[1:-1,1:-1] = -(psi[1:-1, 2:] - psi[1:-1,:-2])/2.0/dx

while err > tol:

psi0 = psi.copy()

omega0 = omega.copy()

psi[1:-1,1:-1] = 0.25\*(psi0[1:-1,2:] + psi0[1:-1,:-2] + psi0[2:,1:-1] + psi0[:-2,1:-1] + dx\*dx\*omega0[1:-1,1:-1])

Cx = -(psi[2:,1:-1] - psi[:-2,1:-1])/2.0/dy \* (omega0[1:-1,2:] - omega0[1:-1,:-2])/2.0/dx

Cy = (psi[1:-1,2:] - psi[1:-1,:-2])/2.0/dx \* (omega0[2:,1:-1] - omega0[:-2,1:-1])/2.0/dy

Dxy = (omega0[1:-1,2:] - 2.0\*omega0[1:-1,1:-1] + omega0[1:-1,:-2])/dx/dx + (omega0[2:,1:-1] -2.0\*omega0[1:-1,1:-1] + omega0[:-2,1:-1])/dy/dy

omega[1:-1,1:-1] = omega0[1:-1,1:-1] + dt\*(Cx + Cy + Nue\*Dxy)

# apply boundary conditions

omega[0,1:-1] = -2.0\*psi[1, 1:-1]/dy/dy # bottom wall

omega[-1,1:-1] = -2.0\*psi[-2, 1:-1]/dy/dy - 2.0\*Ut/dy # vorticity on top wall (moving at Uwall)

omega[1:-1,-1] = -2.0\*psi[1:-1,-2]/dx/dx # right wall

omega[1:-1,0] = -2.0\*psi[1:-1,1]/dx/dx # left wall

err = np.linalg.norm(omega - omega0)

#Plots

solution = omega

plt.contour(solution,levels=200)

plt.colorbar()

plt.xlabel('X')

plt.ylabel('Y')

plt.title("vorticity contours")

plt.show()

u[1:-1,1:-1] = (psi[2:,1:-1] - psi[:-2,1:-1])/2.0/dy

v[1:-1,1:-1] = -(psi[1:-1, 2:] - psi[1:-1,:-2])/2.0/dx

x = np.linspace(0,1,nx)

y = np.linspace(0,1,ny)

xx,yy = np.meshgrid(x,y)

plt.figure(figsize=(5, 5))

plt.streamplot(xx,yy,u, v, color=np.sqrt(u\*u + v\*v),density=4,linewidth=0.5)

plt.xlabel('X')

plt.ylabel('Y')

plt.title('Streamlines')

plt.show()

p = np.zeros([ny,nx])

DxDy = ((omega[1:-1,2:] - 2.0\*omega[1:-1,1:-1] + omega[1:-1,:-2])/dx/dx) \* ((omega[2:,1:-1] -2.0\*omega[1:-1,1:-1] + omega[:-2,1:-1])/dy/dy)

Dxy = (omega[2:,2:] + omega[:-2,:-2] - omega[2:,:-2] - omega[:-2,2:])/4.0/dx/dy

p[1:-1,1:-1] = 0.25\*(p[1:-1,2:] + p[1:-1,:-2] + p[2:,1:-1] + p[:-2,1:-1] - 2\*dx\*dx\*(DxDy - Dxy\*Dxy ))

p[0,1:-1] = p[1,1:-1] # bottom wall

p[-1,1:-1] = p[-2,1:-1] # top wall

p[1:-1,0] = p[1:-1,1] # left wall

p[1:-1,-1] = p[1:-1,-2] # right wall

plt.contourf(xx, yy, p, levels = 100)

plt.colorbar(label='Pressure')

plt.xlabel('X')

plt.ylabel('Y')

plt.title('Pressure')

plt.show()

plt.plot(y, u[:,int(nx/2)], label='U velocity at x=0.5')

plt.legend()

plt.xlabel('Y')

plt.ylabel('U velocity')

plt.title('U velocity at x=0.5')

plt.show()

Uexp = [1.0,0.9766,0.9688,0.9609,0.9531,0.8516,0.7344,0.6172,0.5000,0.4531,0.2813,0.1719,0.1016,0.0703,0.0625,0.0547,0.0000]

yexp = [129,126,125,124,123,110,95,80,65,59,31,23,14,10,9,8,1]

plt.plot(yexp, uexp, label='U velocity at x=0.5')

plt.xlabel('Y')

plt.ylabel('U velocity')

plt.title('U velocity at x=0.5')

plt.show()

plt.plot(y, v[int(ny/2),:], label='V velocity at y=0.5')

plt.legend()

plt.xlabel('X')

plt.ylabel('V velocity')

plt.title('V velocity at y=0.5')

plt.show()

vexp =[1.00,0.9688,0.9609,0.9531,0.9453,0.9063,0.8594,0.8047,0.5000,0.2344,0.2266,0.1563,0.0938,0.0781,0.0703,0.0625,0.0000]

xexp =[129,126,125,124,123,117,111,104,65,31,30,21,13,11,10,9,1]

plt.plot(xexp,vexp, label='Exact solution')

plt.xlabel('X')

plt.ylabel('V velocity')

plt.title('V velocity at y=0.5')

plt.show()