

1. Problem description

This project will involve developing a computer program for two-dimensional unsteady recirculating flows with constant properties (density and viscosity).

Develop a computer program to solve the two-dimensional unsteady Navier-Stokes equations for an elliptic flow in a square cavity with top wall sliding at a given velocity.

The two-dimensional unsteady Navier-Stokes equations for flow with constant density and viscosity are as follows:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \quad (1)$$

$$\rho \left(\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} \right) = - \frac{\partial p}{\partial x} + \mu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \quad (2)$$

$$\rho \left(\frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} \right) = - \frac{\partial p}{\partial y} + \mu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) \quad (3)$$

Use first order time and second-order space discretization. Use an explicit time discretization scheme for convection and implicit scheme for diffusion, and a Pressure-Poisson equation for determining the pressure field at the new time step. Use a collocated scheme for velocities but formulate the pressure equation to avoid checker-board splitting.

After verifying that the code is working, apply the computer program to study the development of the flow in a square cavity with top wall moving. Use any one of your favorite schemes for solving the pressure equation.

Perform calculations for three Reynolds numbers: 20, 50, 100 using grids of 32 x 32 grid intervals (42 x 42 finite difference points). Converge the momentum and pressure Poisson equations by performing 10 and 50 SOR iterations ($\omega = 1.4$) of momentum and pressure respectively at every time step.

Plot the following quantities:

- a) Contours of u and v velocity for the three Reynolds numbers at 4 time intervals separated reasonably well-apart.
- b) Contours of u and v velocity at steady state for the three Reynolds numbers.
- c) Streamlines or velocity vectors for the three Re at steady state.
- d) u-velocity profile on vertical center line and v-velocity profile on horizontal center line for the three Reynolds numbers.
- e) Compare your results with published results in literature and comment on any differences observed.

Write a detailed report with the problem description, equations solved, discretization, numerical algorithm and the results of calculations.

For this project, we set the dimension of square cavity to be $W \times H = 1.0\text{m} \times 1.0\text{m}$, the density of flow to be $\rho = 1\text{kg/m}^3$, the dynamic viscosity to be $\mu = 0.001\text{Pa} \cdot \text{s}$, based on the equation $\text{Re} = \frac{\rho \bar{V}L}{\mu}$, the mean velocity of inlet flow to be 0.02m/s, 0.05m/s and 0.1m/s for $\text{Re} = 20, 50, 100$ respectively.

2. Differencing and Iteration Scheme

Generally, this project was following such a solution sequence:

Define Geometry

Define Grid

Prescribe initial conditions

Prescribe Δt , properties

→ Set up time loop

↑ → r=0

↑ ↑ Solve for $u_{i,j}^{r+1}, v_{i,j}^{r+1}$

↑ ↑ Evaluate $\hat{u}_{i,j}, \hat{v}_{i,j}$

↑ ↑ Interpolate \hat{u}, \hat{v} to half nodes

```

↑ ↑ Evaluate  $S_m$ 
↑ ↑ Evaluate  $A_W^p, A_E^p, A_N^p, A_S^p$ 
↑ ↑ Solve for  $p_{i,j}^{r+1}$ 
↑ ↑ Update  $u_{i,j}^{r+1}, v_{i,j}^{r+1}$ 
↑ ← Go to top of iteration loop
← Go to top of time loop

```

An explicit time discretization scheme for convection and implicit scheme for diffusion were used this project. First order time and second order space discretizations are used in this project. An explicit time discretization scheme has been used for both convection and diffusion. A collocated scheme is used for velocity discretizations and Pressure-Poisson equations are formulated to avoid checker-board splitting and for determining the pressure field at the new time step:

Convection Term:

$$u \frac{\partial u}{\partial x} \Rightarrow \frac{u_{i,j}^n}{2\Delta x} (u_{i+1,j}^n - u_{i-1,j}^n) \quad (4)$$

$$u \frac{\partial v}{\partial x} \Rightarrow \frac{u_{i,j}^n}{2\Delta x} (v_{i+1,j}^n - v_{i-1,j}^n) \quad (5)$$

$$v \frac{\partial u}{\partial y} \Rightarrow \frac{v_{i,j}^n}{2\Delta y} (u_{i,j+1}^n - u_{i,j-1}^n) \quad (6)$$

$$v \frac{\partial v}{\partial y} \Rightarrow \frac{v_{i,j}^n}{2\Delta y} (v_{i,j+1}^n - v_{i,j-1}^n) \quad (7)$$

Diffusion Term:

$$\mu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right)^{n+1} \Rightarrow \frac{\mu}{\Delta x^2} (u_{i+1,j}^{n+1} - 2u_{i,j}^{n+1} + u_{i-1,j}^{n+1}) + \frac{\mu}{\Delta y^2} (u_{i,j+1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j-1}^{n+1}) \quad (8)$$

$$\mu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right)^{n+1} \Rightarrow \frac{\mu}{\Delta x^2} (v_{i+1,j}^{n+1} - 2v_{i,j}^{n+1} + v_{i-1,j}^{n+1}) + \frac{\mu}{\Delta y^2} (v_{i,j+1}^{n+1} - 2v_{i,j}^{n+1} + v_{i,j-1}^{n+1}) \quad (9)$$

Pressure Term:

$$\frac{\partial p}{\partial x} \Rightarrow \frac{p_{i+1,j}^n - p_{i-1,j}^n}{2\Delta x} \quad (10)$$

$$\frac{\partial p}{\partial y} \Rightarrow \frac{p_{i,j+1}^n - p_{i,j-1}^n}{2\Delta y} \quad (11)$$

Other Terms:

$$\frac{\partial u}{\partial t} \Rightarrow \frac{u_{i,j}^{n+1} - u_{i,j}^n}{\Delta t} \quad (12)$$

Rewrite the momentum equation as:

$$A_p^{n+1} u_{i,j}^{n+1} + A_E^{n+1} u_{i+1,j}^{n+1} + A_W^{n+1} u_{i-1,j}^{n+1} + A_N^{n+1} u_{i,j+1}^{n+1} + A_S^{n+1} u_{i,j-1}^{n+1} = RHS \quad (13)$$

where

$$A_p^{n+1} = \frac{\rho}{\Delta t} + \frac{2\mu}{\Delta x^2} + \frac{2\mu}{\Delta y^2} \quad (14)$$

$$A_E^{n+1} = \frac{\rho u_{i,j}^{n+1}}{2\Delta x} - \frac{\mu}{\Delta x^2} \quad (15)$$

$$A_W^{n+1} = -\frac{\rho u_{i,j}^{n+1}}{2\Delta x} - \frac{\mu}{\Delta x^2} \quad (16)$$

$$A_N^{n+1} = \frac{\rho v_{i,j}^{n+1}}{2\Delta y} - \frac{\mu}{\Delta y^2} \quad (17)$$

$$A_S^{n+1} = -\frac{\rho v_{i,j}^{n+1}}{2\Delta y} - \frac{\mu}{\Delta y^2} \quad (18)$$

$$RHS = \frac{\rho u_{i,j}^n}{\Delta t} - \frac{(p_{i+1,j}^{n+1} - p_{i-1,j}^{n+1})}{2\Delta x} \quad (19)$$

Continuity Equation:

For a collocated mesh, we must use half node values:

$$\frac{u_{i+\frac{1}{2},j}^{n+1} - u_{i-\frac{1}{2},j}^{n+1}}{\Delta x} + \frac{v_{i,j+\frac{1}{2}}^{n+1} - v_{i,j-\frac{1}{2}}^{n+1}}{\Delta y} = 0 \quad (20)$$

For the solution of momentum equation at (i, j) , since $p_{i,j}^{n+1}$ is not known, we will use $p_{i,j}^r$,

the value at iteration r for evaluating $(r + 1)$ values at time step $(n + 1)$.

Evaluate $\hat{u}_{i,j}, \hat{v}_{i,j}$. Define an interpolation scheme, define momentum velocities as:

$$\hat{u}_{i,j} = u_{i,j}^{r+1} + \frac{1}{(A_p)_{i,j}} \frac{(p_{i+1,j}^{r+1} - p_{i-1,j}^{r+1})}{2\Delta x} \quad (21)$$

$$\hat{v}_{i,j} = v_{i,j}^{r+1} + \frac{1}{(A_p)_{i,j}} \frac{(p_{i,j+1}^{r+1} - p_{i,j-1}^{r+1})}{2\Delta y} \quad (22)$$

Define the $(r+1)$ velocities at $\left(i + \frac{1}{2}, j\right)$, $\left(i - \frac{1}{2}, j\right)$, $\left(i, j + \frac{1}{2}\right)$, $\left(i, j - \frac{1}{2}\right)$ points as:

$$u_{i+\frac{1}{2},j}^{r+1} = \hat{u}_{i+\frac{1}{2},j}^{r+1} + \frac{1}{(A_p)_{i+\frac{1}{2},j}} \frac{(p_{i,j}^{r+1} - p_{i+1,j}^{r+1})}{\Delta x} \quad (23)$$

$$u_{i-\frac{1}{2},j}^{r+1} = \hat{u}_{i-\frac{1}{2},j}^{r+1} + \frac{1}{(A_p)_{i-\frac{1}{2},j}} \frac{(p_{i-1,j}^{r+1} - p_{i,j}^{r+1})}{\Delta x} \quad (24)$$

$$v_{i,j+\frac{1}{2}}^{r+1} = \hat{v}_{i,j+\frac{1}{2}}^{r+1} + \frac{1}{(A_p)_{i,j+\frac{1}{2}}} \frac{(p_{i,j}^{r+1} - p_{i,j+1}^{r+1})}{\Delta y} \quad (25)$$

$$v_{i,j-\frac{1}{2}}^{r+1} = \hat{v}_{i,j-\frac{1}{2}}^{r+1} + \frac{1}{(A_p)_{i,j-\frac{1}{2}}} \frac{(p_{i,j-1}^{r+1} - p_{i,j}^{r+1})}{\Delta y} \quad (26)$$

Evaluate $\hat{u}_{i,j}$, $\hat{v}_{i,j}$ at midpoints:

$$\hat{u}_{i+\frac{1}{2},j}^{r+1} = \frac{1}{2} (\hat{u}_{i,j} + \hat{u}_{i+1,j})^{r+1} \quad (27)$$

$$\hat{u}_{i-\frac{1}{2},j}^{r+1} = \frac{1}{2} (\hat{u}_{i,j} + \hat{u}_{i-1,j})^{r+1} \quad (28)$$

$$\hat{v}_{i,j+\frac{1}{2}}^{r+1} = \frac{1}{2} (\hat{v}_{i,j} + \hat{v}_{i,j+1})^{r+1} \quad (29)$$

$$\hat{v}_{i,j-\frac{1}{2}}^{r+1} = \frac{1}{2} (\hat{v}_{i,j} + \hat{v}_{i,j-1})^{r+1} \quad (30)$$

For the Pressure Poisson Equation, we use SOR Iteration scheme to converge the equation, omega = 1.4 was adopted here. Run 50 and 10 SOR Iteration for momentum equation and PPE respectively at each time step.

Derivation of Pressure Equation:

$$\frac{u_{i+\frac{1}{2},j} - u_{i-\frac{1}{2},j}}{\Delta x} + \frac{v_{i,j+\frac{1}{2}} - v_{i,j-\frac{1}{2}}}{\Delta y} = 0 \quad (31)$$

Write this in terms of pressures and \hat{u} :

$$\begin{aligned} & \left[\left\{ \hat{u}_{i+\frac{1}{2},j} + \frac{1}{(A_p)_{i+\frac{1}{2},j}} \frac{(p_{i,j}^{r+1} - p_{i+1,j}^{r+1})}{\Delta x} \right\} - \left\{ \hat{u}_{i-\frac{1}{2},j} + \frac{1}{(A_p)_{i-\frac{1}{2},j}} \frac{(p_{i-1,j}^{r+1} - p_{i,j}^{r+1})}{\Delta x} \right\} \right] / \Delta x \\ & + \left[\left\{ \hat{v}_{i,j+\frac{1}{2}} + \frac{1}{(A_p)_{i,j+\frac{1}{2}}} \frac{(p_{i,j}^{r+1} - p_{i,j+1}^{r+1})}{\Delta y} \right\} - \left\{ \hat{v}_{i,j-\frac{1}{2}} + \frac{1}{(A_p)_{i,j-\frac{1}{2}}} \frac{(p_{i,j-1}^{r+1} - p_{i,j}^{r+1})}{\Delta y} \right\} \right] / \Delta y \\ & = 0 \end{aligned} \quad (32)$$

Rearrange this with pressures on one side and mass errors on the other side. LHS can be

written as:

$$\begin{aligned}
& p_{i,j}^{r+1} \left(\frac{1}{(A_p)_{i+\frac{1}{2},j}} \frac{1}{\Delta x^2} + \frac{1}{(A_p)_{i-\frac{1}{2},j}} \frac{1}{\Delta x^2} + \frac{1}{(A_p)_{i,j+\frac{1}{2}}} \frac{1}{\Delta y^2} + \frac{1}{(A_p)_{i,j-\frac{1}{2}}} \frac{1}{\Delta y^2} \right) \\
&= p_{i+1,j}^{r+1} \left(\frac{1}{(A_p)_{i+\frac{1}{2},j}} \frac{1}{\Delta x^2} \right) + p_{i-1,j}^{r+1} \left(\frac{1}{(A_p)_{i-\frac{1}{2},j}} \frac{1}{\Delta x^2} \right) \\
&+ p_{i,j+1}^{r+1} \left(\frac{1}{(A_p)_{i,j+\frac{1}{2}}} \frac{1}{\Delta y^2} \right) + p_{i,j-1}^{r+1} \left(\frac{1}{(A_p)_{i,j-\frac{1}{2}}} \frac{1}{\Delta y^2} \right) - \frac{\hat{u}_{i+\frac{1}{2},j} - \hat{u}_{i-\frac{1}{2},j}}{\Delta x} \\
&+ \frac{\hat{v}_{i,j+\frac{1}{2}} - \hat{v}_{i,j-\frac{1}{2}}}{\Delta y}
\end{aligned} \tag{33}$$

This can be written as:

$$(A_p^p)_{i,j} p_{i,j}^{r+1} = (A_E^p)_{i,j} p_{i+1,j}^{r+1} + (A_W^p)_{i,j} p_{i-1,j}^{r+1} + (A_N^p)_{i,j} p_{i,j+1}^{r+1} + (A_S^p)_{i,j} p_{i,j-1}^{r+1} + S_m \tag{34}$$

where

$$A_p^p = A_W^p + A_E^p + A_N^p + A_S^p \tag{35}$$

$A_W^p, A_E^p, A_N^p, A_S^p$ are given as equation (12) (13) (14) (15).

Evaluate A_p by interpolation:

$$\frac{1}{(A_p)_{i+\frac{1}{2},j}} = \frac{1}{2} \left(\frac{1}{(A_p)_{i,j}} + \frac{1}{(A_p)_{i+1,j}} \right) \tag{36}$$

$$\frac{1}{(A_p)_{i-\frac{1}{2},j}} = \frac{1}{2} \left(\frac{1}{(A_p)_{i,j}} + \frac{1}{(A_p)_{i-1,j}} \right) \tag{37}$$

$$\frac{1}{(A_p)_{i,j+\frac{1}{2}}} = \frac{1}{2} \left(\frac{1}{(A_p)_{i,j}} + \frac{1}{(A_p)_{i,j+1}} \right) \tag{38}$$

$$\frac{1}{(A_p)_{i,j-\frac{1}{2}}} = \frac{1}{2} \left(\frac{1}{(A_p)_{i,j}} + \frac{1}{(A_p)_{i,j-1}} \right) \tag{39}$$

Update $u_{i,j}^{r+1}, v_{i,j}^{r+1}$:

$$u_{i,j}^{n+1} = \hat{u}_{i,j} - \frac{\Delta t}{\rho} \left(\frac{p_{i+1,j}^{n+1} - p_{i-1,j}^{n+1}}{2\Delta x} \right) \tag{40}$$

For stability issues, since fully explicit scheme was used in this project, the time step size

Δt yields to the following stability limit:

$$\Delta t \leq \frac{1}{\frac{|u|}{\Delta x} + \frac{|v|}{\Delta y} + \frac{2\nu}{\Delta x^2} + \frac{2\nu}{\Delta y^2}} \quad (41)$$

where $\Delta x = \Delta y = 0.025m$, kinematic viscosity $\nu = \mu/\rho = 0.001m^2 \cdot s$, $u = 0.02, 0.05, 0.1m/s$ for $Re = 20, 50, 100$, respectively, $v = 0$ for all three Reynolds numbers.

So the critical time step size $\Delta t = 0.1388s, 0.1190s$ and $0.096s$, for $Re = 20, 50, 100$, respectively.

3. Geometry, Boundary Conditions and Initial Conditions

For this project, we set the dimension of square cavity to be $W \times H = 1.0m \times 1.0m$, the density of flow to be $\rho = 1kg/m^3$, the dynamic viscosity to be $\mu = 0.001Pa \cdot s$.

For boundary conditions, the velocities at all walls expect top wall were all set to be 0.

For initial conditions, based on the equation $Re = \frac{\rho \bar{V}L}{\mu}$, the mean u-velocity of flow at top wall was to be $0.02m/s, 0.05m/s$ and $0.1m/s$ for $Re = 20, 50, 100$ respectively, v-velocity was set to be 0.

Also it should be noticed that since we evaluated $u_{i,j}, v_{i,j}$ at midpoints, ghost nodes located at outside the domain of interest should be introduced to ensure that the initial conditions and boundary conditions were met.

4. Results and Discussion

Figure 1 shows the contours of u velocity for $Re=20$ at four time intervals.

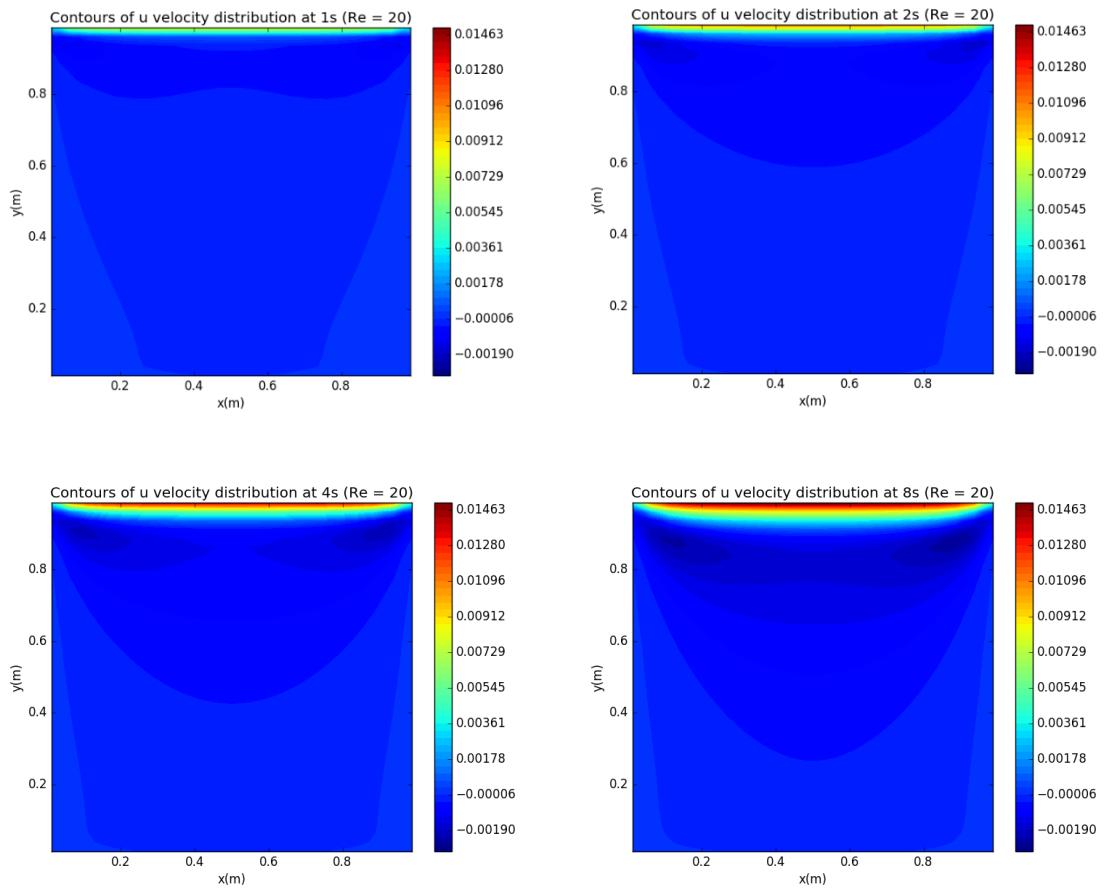
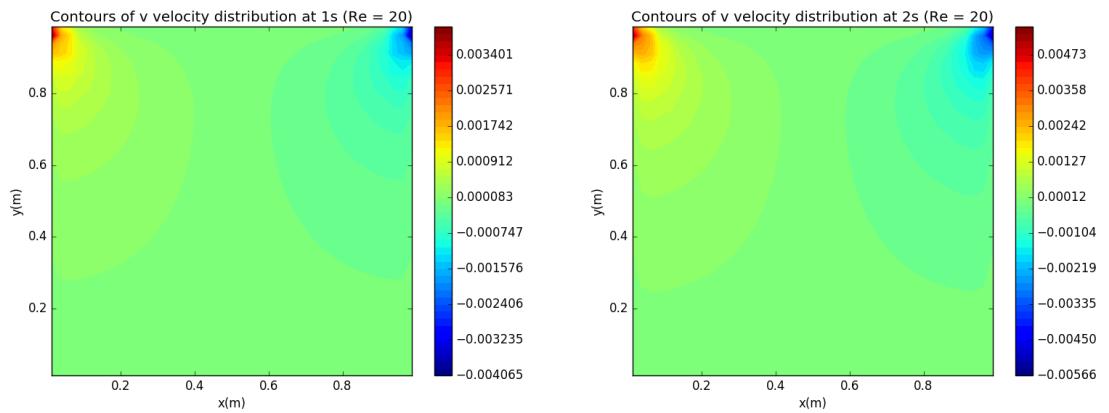


Figure 1. Contours of u-velocity at four different time intervals (Re=20)

Figure 2 shows the contours of v velocity for Re=20 at four time intervals.



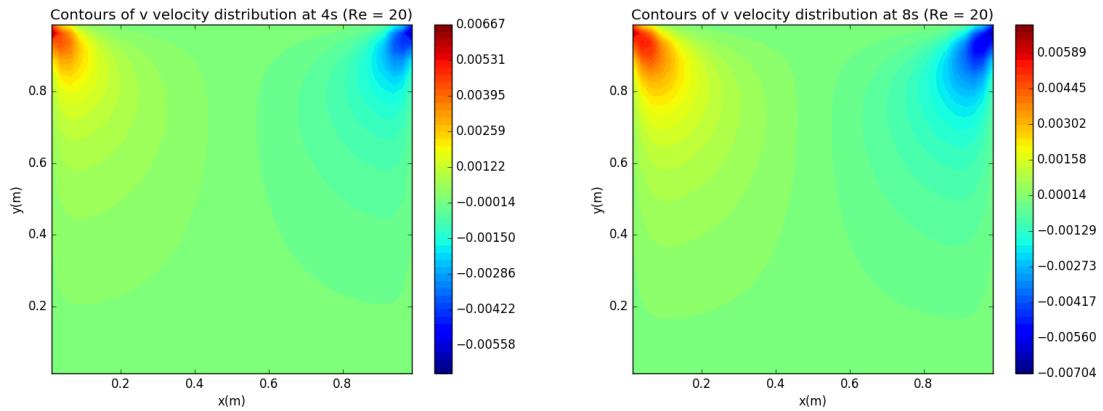


Figure 2. Contours of v-velocity at four different time intervals (Re=20)

Figure 3 shows the contours of u velocity for Re=50 at four time intervals.

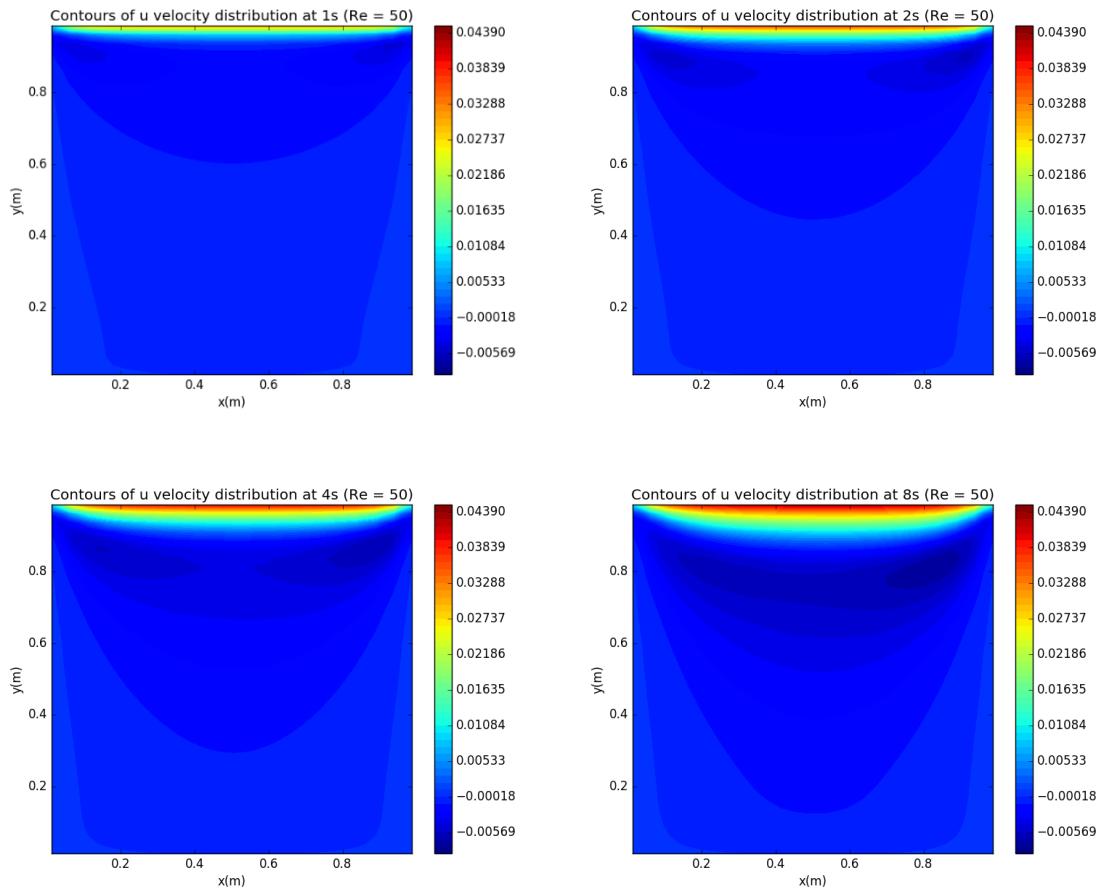


Figure 3. Contours of u-velocity at four different time intervals (Re=50)

Figure 4 shows the contours of v velocity for Re=50 at four time intervals.

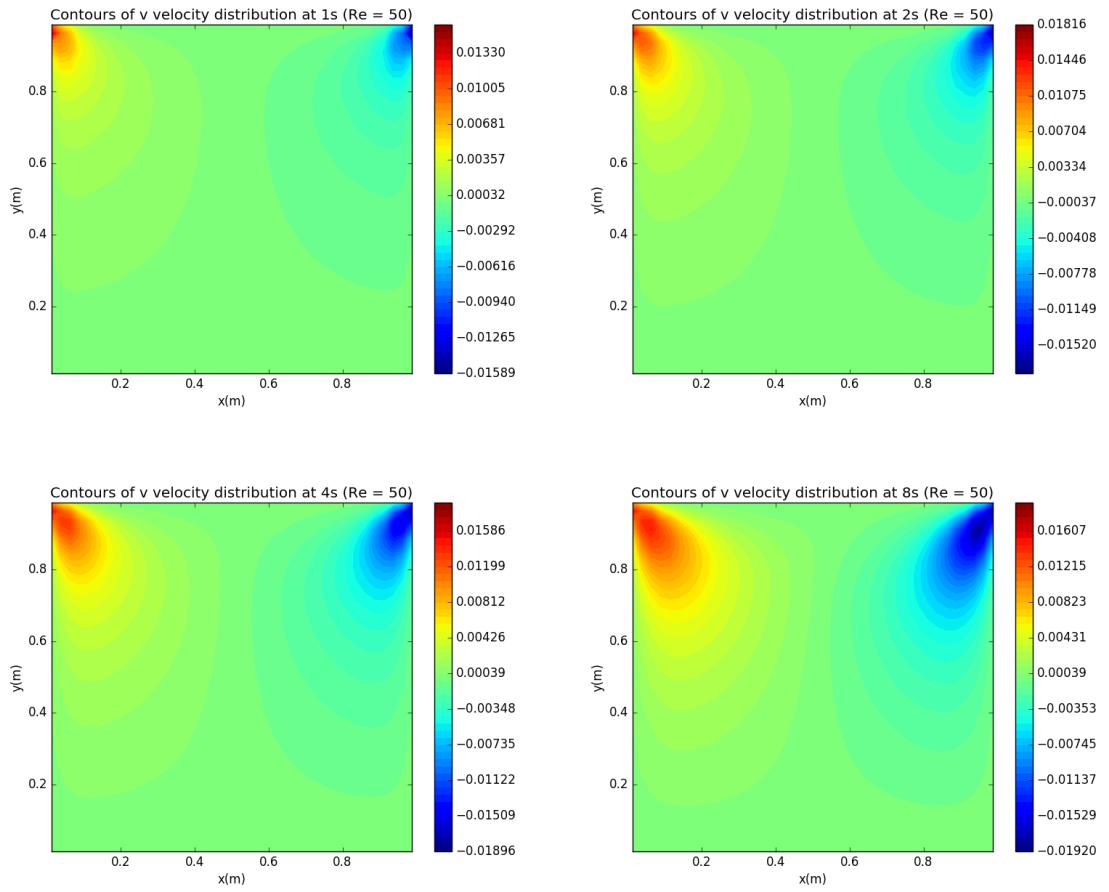
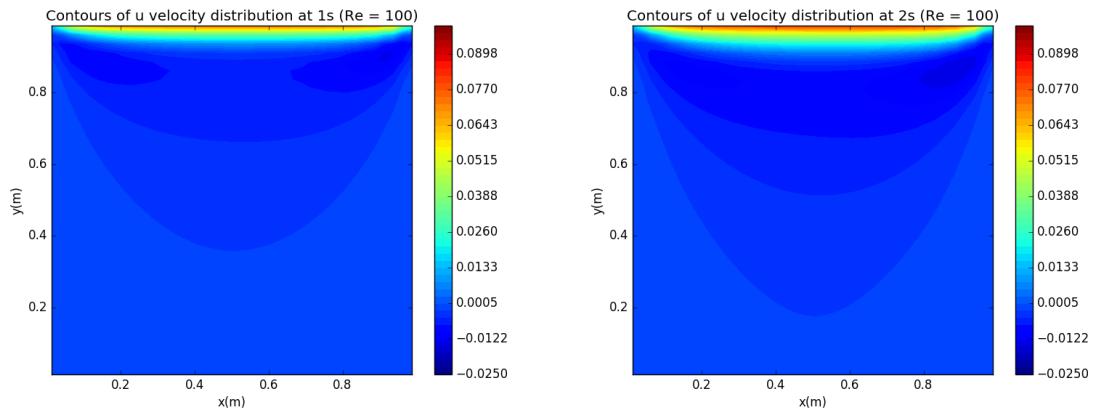


Figure 4. Contours of v-velocity at four different time intervals (Re=50)

Figure 5 shows the contours of u velocity for Re=100 at four time intervals.



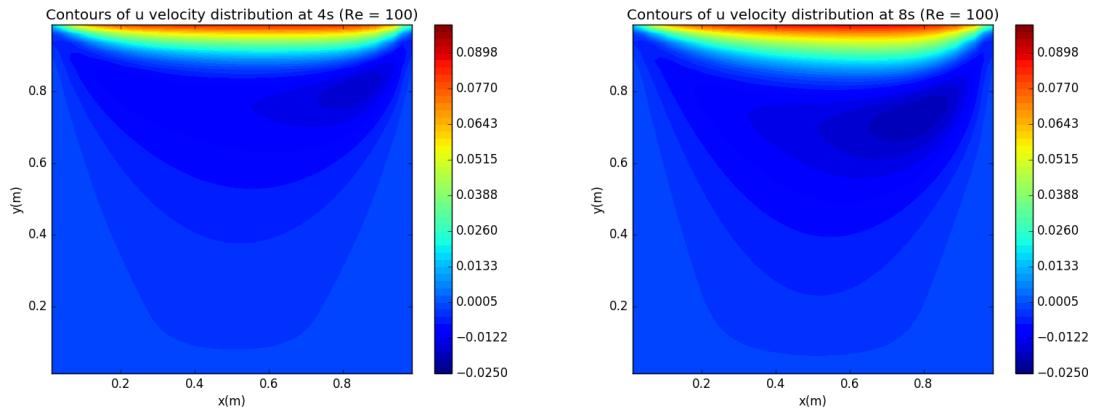


Figure 5. Contours of u-velocity at four different time intervals (Re=100)

Figure 6 shows the contours of v velocity for Re=100 at four time intervals.

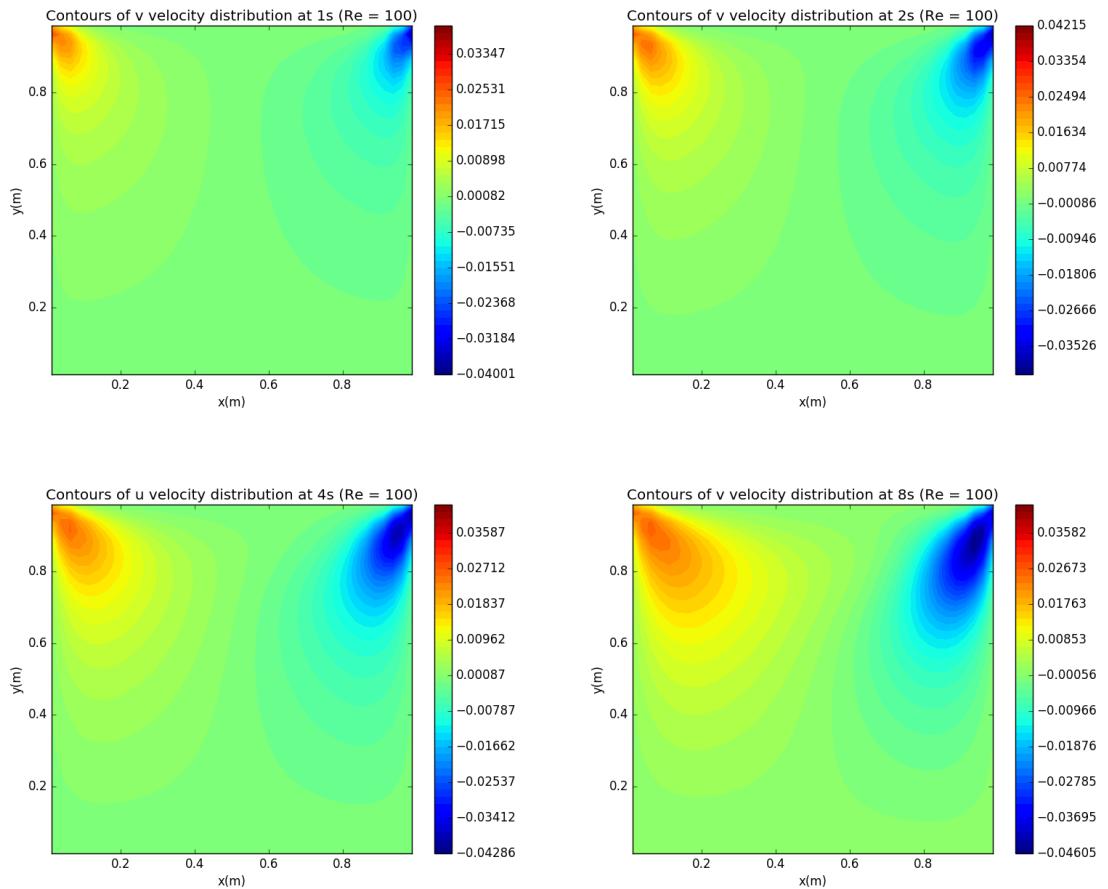
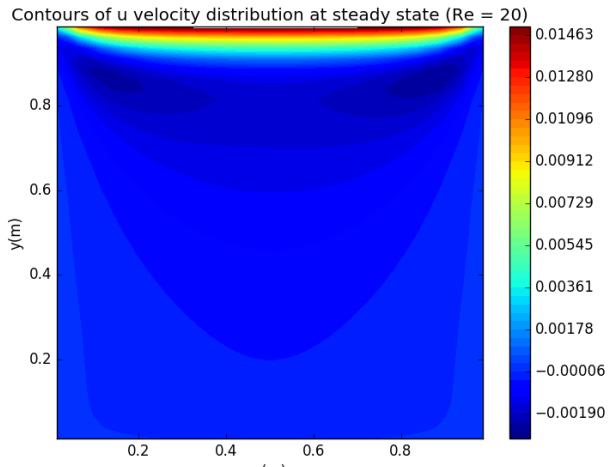
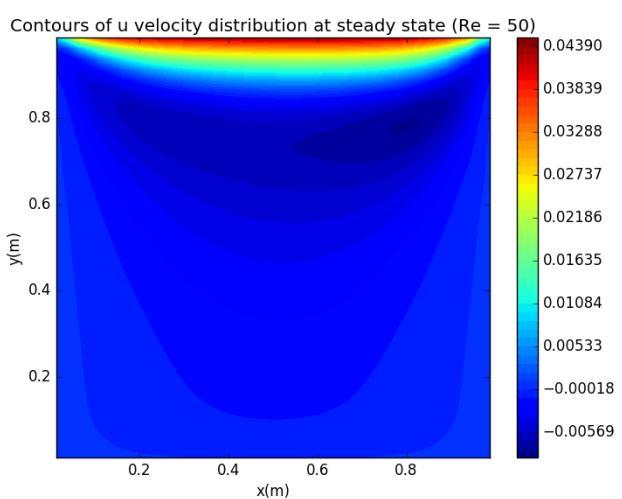


Figure 6. Contours of v-velocity at four different time intervals (Re=100)

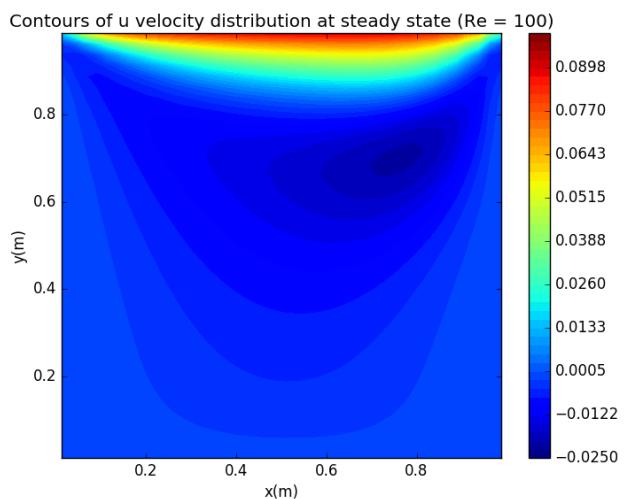
Figure 7(a)(b)(c) shows a comparison of contours of u velocity at steady state for different Reynolds numbers. And Figure 8(a)(b)(c) shows a comparison of contours of v velocity at steady state for different Reynolds numbers.



(a) Re=20

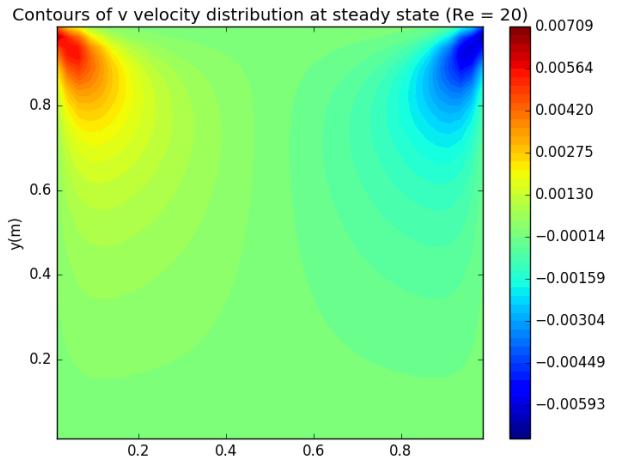


(b) Re=50

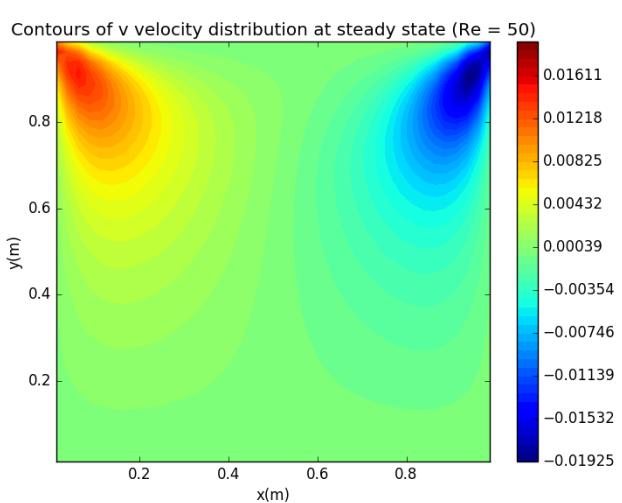


(c) Re=100

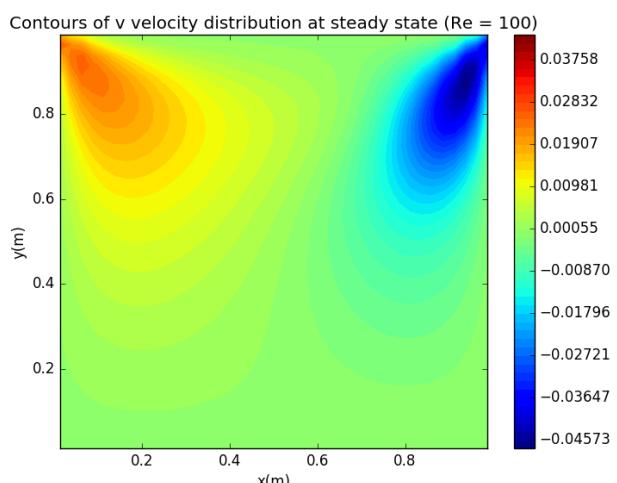
Figure 7. Contours of u-velocity at steady state for three different Reynolds numbers



(a) Re=20



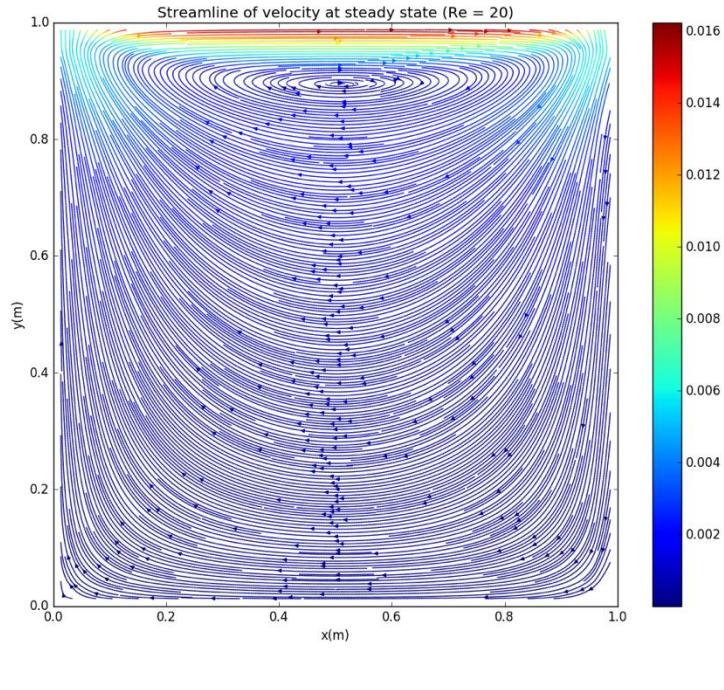
(b) Re=50



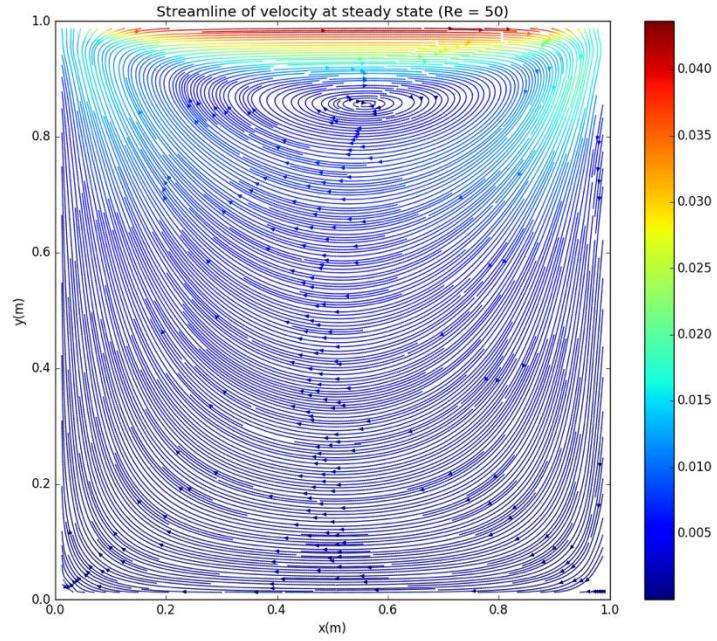
(c) Re=100

Figure 8. Contours of v-velocity at steady state for three different Reynolds numbers

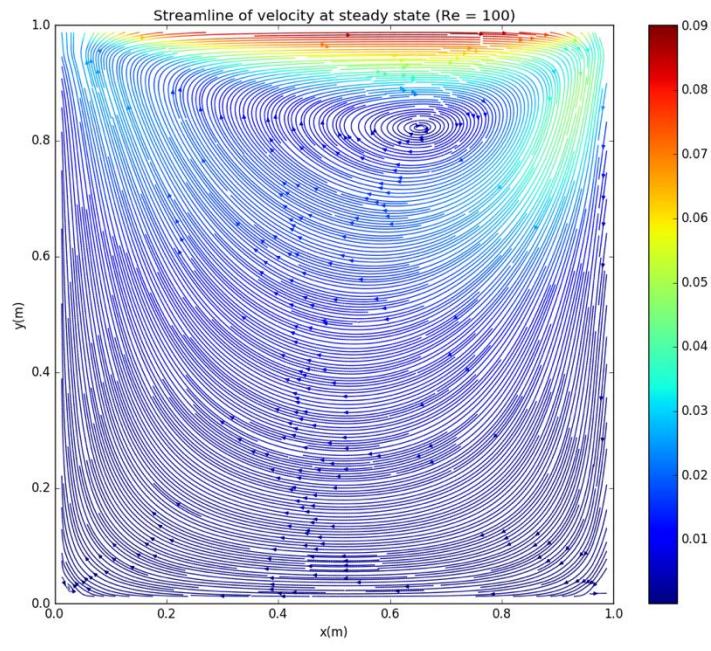
Figure 9 shows the streamlines of velocity vector for three different Reynolds numbers at steady state. It can be observed that there was a vortex formed inside the square cavity and its center moved downward as the Reynolds number increased. The rotational direction of the vortex is clockwise, which is in accord with the moving direction of the top wall.



(a) Re=20



(b) Re=50



(c) $Re=100$

Figure 9. Streamlines of velocity vectors at steady state for three different Reynolds numbers

Figure 10 shows the u-velocity profile on vertical center line for three Reynolds numbers.

Figure 11 shows the v-velocity profile on horizontal center line for three Reynolds numbers.

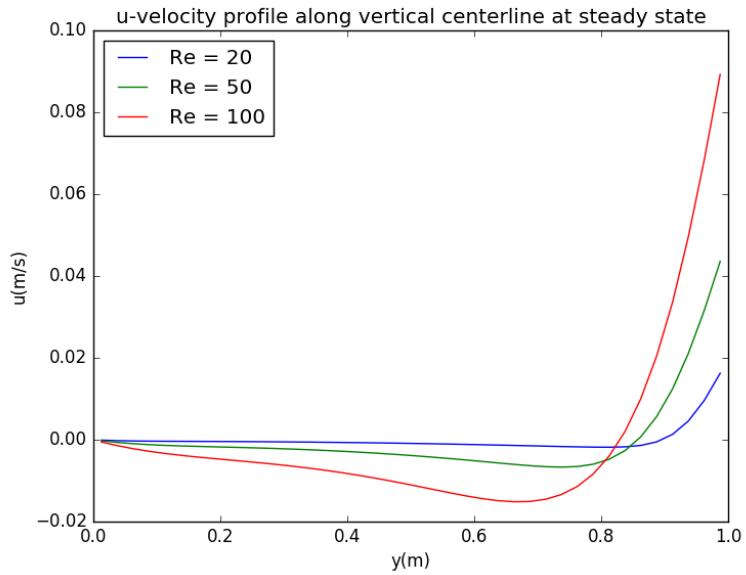


Figure 10. u-velocity profile on vertical center line for three Reynolds numbers

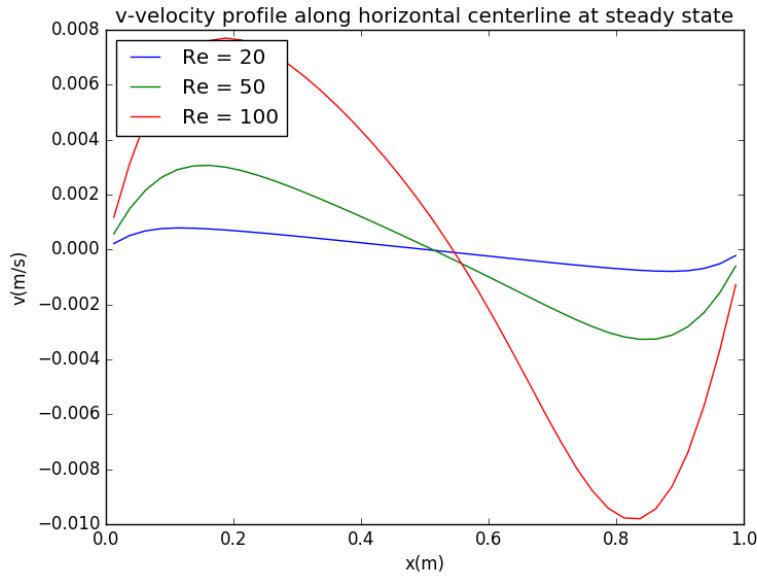


Figure 11. v-velocity profile on horizontal center line for three Reynolds numbers

In Figure 12 the profiles of u velocity at the vertical centerline generated for $Re=100$ at steady state are compared with those generated by Ekebjærg and Justesenu [1]. They are in good agreement with each other.

In Figure 15 the streamlines of velocity vector we generated for $Re=100$ at steady state are compared with those generated by Ekebjærg and Justesenu [1]. There was a little difference that in our plot we could not observe the potential vortices at bottom left and bottom right corners, however, in

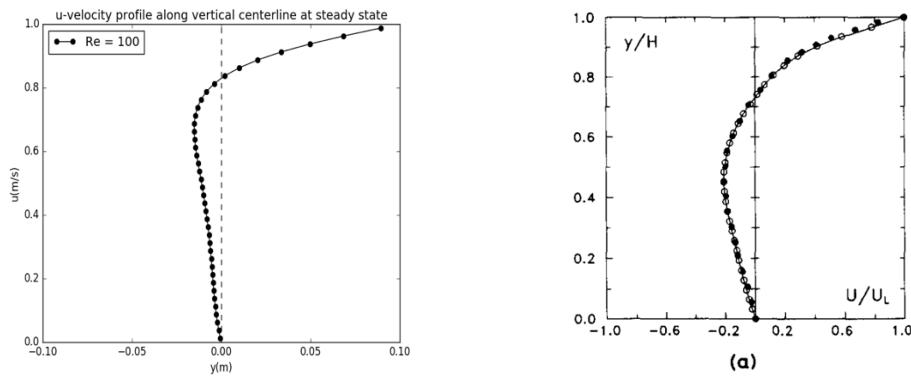


Figure 12. Comparison of u-velocity profile on vertical centerline with published results in literature ($Re=100$) (Figure 12(b) adapted from Ekebjærg, L., & Justesenu, P. (1991). An explicit scheme for advection-diffusion modelling in two dimensions. Computer Methods in Applied Mechanics and Engineering, 88(3), 287-297.)

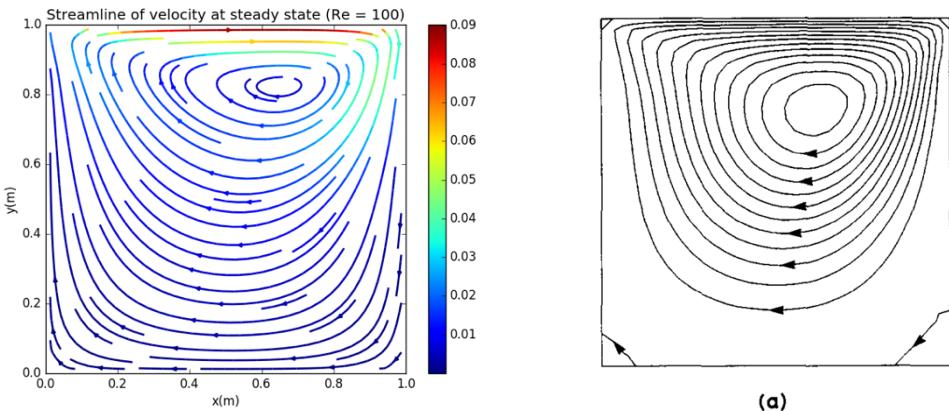


Figure 13. Comparison of streamlines of velocity vector with published results in literature (Re=100) (Figure 13(b) adapted from Ekebjærg, L., & Justesenu, P. (1991). An explicit scheme for advection-diffusion modelling in two dimensions. Computer Methods in Applied Mechanics and Engineering, 88(3), 287-297.)

5. Conclusions

In this project, we successfully developed a program to solve the two-dimensional unsteady Navier- Stokes equations for an elliptic flow in a square cavity with top wall sliding at a given velocity. Numerical results for flow of different Reynolds numbers at different time intervals and steady state have been shown. Satisfactory agreement with results from other literatures has been achieved.

Reference

- [1] Ekebjærg, L., & Justesenu, P. (1991). An explicit scheme for advection-diffusion modelling in two dimensions. Computer Methods in Applied Mechanics and Engineering, 88(3), 287-297.

Appendix: Source Code

Project5.py

```
# 2D - Lid Driven Cavity Flow with Constant Density and Viscosity
# 2D - Unsteady Navier-Stokes equations for an elliptic flow

# Explicit time discretization scheme for convection
# Implicit time discretization scheme for diffusion
# Poisson Pressure Equation(PPE) for determining the pressure at new time step
# Collocated scheme for velocities

import numpy as np
import matplotlib.pyplot as plt
import pylab

import plotly.plotly as py
import plotly.figure_factory as ff
import plotly.tools as tls
tls.set_credentials_file(username='king_yin3613', api_key='cELFEragb4loopx5xzxx')

# Parameters
rho = 1.0
mu = 0.001
nu = mu/rho
Re = 20
Re_1 = 50
Re_2 = 100
omega = 1.4

# Geometry
x_min = 0.0
x_max = 1.0
Lx = x_max - x_min

y_min = 0.0
y_max = 1.0
Ly = y_max - y_min

# Grid
x_node = 42
y_node = 42
```

```

dx = (x_max - x_min)/(x_node - 2)
dy = (y_max - y_min)/(y_node - 2)

x_grid = np.arange(x_min, x_max+dx, dx)
x_grid_half = np.arange(x_min+dx/2, x_max,dx)
y_grid = np.arange(y_min, y_max+dy, dy)
y_grid_half = np.arange(y_min+dy/2, y_max,dy)

# Initial Conditions
v_ini = Re * mu / (rho * Lx)

# Time-step
dt_limit = 1.0/(v_ini/dx+2*(nu/(dx**2)+nu/(dy**2)))
print('dt_limit = %s' % dt_limit)
dt = 0.05
n_momentum = 10
n_SOR = 50
timestep = int(100/dt)

# Solver
def Solve(Re,time):

    dx = (x_max - x_min)/(x_node - 2)
    dy = (y_max - y_min)/(y_node - 2)

    # Initial conditions
    u_old = np.zeros([y_node,x_node])
    v_old = np.zeros([y_node,x_node])
    u_new = np.zeros([y_node,x_node])
    v_new = np.zeros([y_node,x_node])
    p_old = np.zeros([y_node,x_node])
    p_new = np.zeros([y_node,x_node])
    uhat = np.zeros([y_node,x_node])
    vhat = np.zeros([y_node,x_node])
    uhat_midpoint = np.zeros([y_node,x_node-1])
    vhat_midpoint = np.zeros([y_node-1,x_node])
    error = np.zeros([y_node,x_node])
    for j in range(1,x_node-1):
        u_old[-1,j] = 2*v_ini
    #print('u = %s' % u_old)

    for n in range(1,int(time/dt)+1):

```

```

for Momentum_count in range(0,n_momentum):
    # Evaluate uhat and vhat
    for i in range(1,y_node-1):
        for j in range(1,x_node-1):
            C_x = rho*u_old[i,j]/(2*dx)*(u_old[i,j+1]-u_old[i,j-1])+rho*v_old[i,j]/(2*dy)*(u_old[i+1,j]-u_old[i-1,j])
            C_y = rho*u_old[i,j]/(2*dx)*(v_old[i,j+1]-v_old[i,j-1])+rho*v_old[i,j]/(2*dy)*(v_old[i+1,j]-v_old[i-1,j])
            D_x = mu/(dx**2)*(u_old[i,j+1]-2*u_old[i,j]+u_old[i,j-1])+mu/(dy**2)*(u_old[i+1,j]-2*u_old[i,j]+u_old[i-1,j])
            D_y = mu/(dx**2)*(v_old[i,j+1]-2*v_old[i,j]+v_old[i,j-1])+mu/(dy**2)*(v_old[i+1,j]-2*v_old[i,j]+v_old[i-1,j])
            uhat[i,j] = u_old[i,j]+dt/rho*(-C_x+D_x);
            vhat[i,j] = v_old[i,j]+dt/rho*(-C_y+D_y);

    # Evaluate uhat and vhat at midpoints
    for i in range(0,y_node):
        for j in range(1,x_node-2):
            uhat_midpoint[i,j] = (uhat[i,j]+uhat[i,j+1])/2

    for i in range(1,y_node-2):
        for j in range(0,x_node):
            vhat_midpoint[i,j] = (vhat[i,j]+uhat[i+1,j])/2

    for i in range(1,y_node-1):
        for j in range(1,x_node-1):
            error[i,j] = rho/dt*((uhat_midpoint[i,j]-uhat_midpoint[i,j-1])/dx+(vhat_midpoint[i,j]-vhat_midpoint[i-1,j])/dy)

    # SOR Iteration to solve PPE (omega = 1.4, n = 50)
    p_new = np.copy(p_old)

    for SOR_count in range(0,n_SOR):
        for j in range(1,y_node-1):
            p_new[0,j] = p_new[1,j]
            p_new[-1,j] = p_new[-2,j]
        for i in range(1,x_node-1):
            p_new[i,0] = p_new[i,1]
            p_new[i,-1] = p_new[i,-2]

        for i in range(1,y_node-1):
            for j in range(1,x_node-1):

```

```

R = ((p_new[i,j+1]+p_new[i,j-1])/(dx**2)+(p_new[i+1,j]+p_new[i-1,j])/((dy**2)-error[i,j]))/(2/(dx**2)+2/(dy**2))
p_new[i,j] = omega*R + (1-omega)*p_new[i,j]

for i in range(1,y_node-1):
    p_new[i,0] = p_new[i,1]
    p_new[i,-1] = p_new[i,-2]
for j in range(1,x_node-1):
    p_new[0,j] = p_new[1,j]
    p_new[-1,j] = p_new[-2,j]

# Correct grid node velocities
for i in range(1,y_node-1):
    for j in range(1,x_node-1):
        u_new[i,j] = uhat[i,j]-dt/rho*(p_new[i,j+1]-p_new[i,j-1])/(2*dx)
        v_new[i,j] = vhat[i,j]-dt/rho*(p_new[i+1,j]-p_new[i-1,j])/(2*dy)

# Update ghost node values
for j in range(1,x_node-1):
    u_new[-1,j] = 2*v_ini - u_new[-2,j]
    u_new[0,j] = -u_new[1,j]
    v_new[-1,j] = -v_new[-2,j]
    v_new[0,j] = -v_new[1,j]
for i in range(1,y_node-1):
    u_new[i,-1] = -u_new[i,-2]
    u_new[i,0] = -u_new[i,1]
    v_new[i,-1] = -v_new[i,-2]
    v_new[i,0] = -v_new[i,1]

u_old = np.copy(u_new)
v_old = np.copy(v_new)
p_old = np.copy(p_new)

return u_new,v_new,p_new

# Re = 20
Re_20_u_1s = Solve(Re,1.0)[0]
Re_20_u_2s = Solve(Re,2.0)[0]
Re_20_u_4s = Solve(Re,4.0)[0]
Re_20_u_8s = Solve(Re,8.0)[0]
Re_20_u_steady = Solve(Re,10.0)[0]
Re_20_v_1s = Solve(Re,1.0)[1]

```

```
Re_20_v_2s = Solve(Re,2.0)[1]
Re_20_v_4s = Solve(Re,4.0)[1]
Re_20_v_8s = Solve(Re,8.0)[1]
Re_20_v_steady = Solve(Re,10.0)[1]
```

```
# Re = 50
Re_50_u_1s = Solve(Re_1,1.0)[0]
Re_50_u_2s = Solve(Re_1,2.0)[0]
Re_50_u_4s = Solve(Re_1,4.0)[0]
Re_50_u_8s = Solve(Re_1,8.0)[0]
Re_50_u_steady = Solve(Re_1,10.0)[0]
Re_50_v_1s = Solve(Re_1,1.0)[1]
Re_50_v_2s = Solve(Re_1,2.0)[1]
Re_50_v_4s = Solve(Re_1,4.0)[1]
Re_50_v_8s = Solve(Re_1,8.0)[1]
Re_50_v_steady = Solve(Re_1,10.0)[1]
```

```
# Re = 100
Re_100_u_1s = Solve(Re_2,1.0)[0]
Re_100_u_2s = Solve(Re_2,2.0)[0]
Re_100_u_4s = Solve(Re_2,4.0)[0]
Re_100_u_8s = Solve(Re_2,8.0)[0]
Re_100_u_steady = Solve(Re_2,10.0)[0]
Re_100_v_1s = Solve(Re_2,1.0)[1]
Re_100_v_2s = Solve(Re_2,2.0)[1]
Re_100_v_4s = Solve(Re_2,4.0)[1]
Re_100_v_8s = Solve(Re_2,8.0)[1]
Re_100_v_steady = Solve(Re_2,10.0)[1]
```

```
# Visualization
plt.figure(11)
levels = np.linspace(-0.025,0.10,50)
CS = plt.contourf(x_grid_half, y_grid_half, Re_100_u_1s[1:y_node-1,1:x_node-1],levels=levels)
plt.title('Contours of u velocity distribution at 1s (Re = 100)')
pylab.xlabel('x(m)')
pylab.ylabel('y(m)')
pylab.colorbar(CS)
plt.show()
```

```
plt.figure(12)
levels = np.linspace(-0.025,0.10,50)
```

```
CS = plt.contourf(x_grid_half, y_grid_half, Re_100_u_2s[1:y_node-1,1:x_node-1],levels=levels)
```

```
plt.title('Contours of u velocity distribution at 2s (Re = 100)')
```

```
pylab.xlabel('x(m)')
```

```
pylab.ylabel('y(m)')
```

```
pylab.colorbar(CS)
```

```
plt.show()
```

```
plt.figure(13)
```

```
levels = np.linspace(-0.025,0.10,50)
```

```
CS = plt.contourf(x_grid_half, y_grid_half, Re_100_u_4s[1:y_node-1,1:x_node-1],levels=levels)
```

```
plt.title('Contours of u velocity distribution at 4s (Re = 100)')
```

```
pylab.xlabel('x(m)')
```

```
pylab.ylabel('y(m)')
```

```
pylab.colorbar(CS)
```

```
plt.show()
```

```
plt.figure(14)
```

```
levels = np.linspace(-0.025,0.10,50)
```

```
CS = plt.contourf(x_grid_half, y_grid_half, Re_100_u_8s[1:y_node-1,1:x_node-1],levels=levels)
```

```
plt.title('Contours of u velocity distribution at 8s (Re = 100)')
```

```
pylab.xlabel('x(m)')
```

```
pylab.ylabel('y(m)')
```

```
pylab.colorbar(CS)
```

```
plt.show()
```

```
plt.figure(15)
```

```
levels = np.linspace(-0.025,0.10,50)
```

```
CS = plt.contourf(x_grid_half, y_grid_half, Re_100_u_steady[1:y_node-1,1:x_node-1],levels=levels)
```

```
plt.title('Contours of u velocity distribution at steady state (Re = 100)')
```

```
pylab.xlabel('x(m)')
```

```
pylab.ylabel('y(m)')
```

```
pylab.colorbar(CS)
```

```
plt.show()
```

```
plt.figure(21)
```

```
levels = np.linspace(np.amin(Re_100_v_1s),np.amax(Re_100_v_1s),50)
```

```
CS = plt.contourf(x_grid_half, y_grid_half, Re_100_v_1s[1:y_node-1,1:x_node-1],levels=levels)
```

```

plt.title('Contours of v velocity distribution at 1s (Re = 100)')
pylab.xlabel('x(m)')
pylab.ylabel('y(m)')
pylab.colorbar(CS)
plt.show()

plt.figure(22)
levels = np.linspace(np.amin(Re_100_v_2s),np.amax(Re_100_v_2s),50)
CS = plt.contourf(x_grid_half, y_grid_half, Re_100_v_2s[1:y_node-1,1:x_node-1],levels=levels)
plt.title('Contours of v velocity distribution at 2s (Re = 100)')
pylab.xlabel('x(m)')
pylab.ylabel('y(m)')
pylab.colorbar(CS)
plt.show()

plt.figure(23)
levels = np.linspace(np.amin(Re_100_v_4s),np.amax(Re_100_v_4s),50)
CS = plt.contourf(x_grid_half, y_grid_half, Re_100_v_4s[1:y_node-1,1:x_node-1],levels=levels)
plt.title('Contours of v velocity distribution at 4s (Re = 100)')
pylab.xlabel('x(m)')
pylab.ylabel('y(m)')
pylab.colorbar(CS)
plt.show()

plt.figure(24)
levels = np.linspace(np.amin(Re_100_v_8s),np.amax(Re_100_v_8s),50)
CS = plt.contourf(x_grid_half, y_grid_half, Re_100_v_8s[1:y_node-1,1:x_node-1],levels=levels)
plt.title('Contours of v velocity distribution at 8s (Re = 100)')
pylab.xlabel('x(m)')
pylab.ylabel('y(m)')
pylab.colorbar(CS)
plt.show()

plt.figure(25)
levels = np.linspace(np.amin(Re_100_v_steady),np.amax(Re_100_v_steady),50)
CS = plt.contourf(x_grid_half, y_grid_half, Re_100_v_steady[1:y_node-1,1:x_node-1],levels=levels)
plt.title('Contours of v velocity distribution at steady state (Re = 100)')
pylab.xlabel('x(m)')

```

```

pylab.ylabel('y(m)')
pylab.colorbar(CS)
plt.show()

plt.figure(31)
levels = np.linspace(-0.003,0.015,50)
CS = plt.contourf(x_grid_half, y_grid_half, Re_20_u_1s[1:y_node-1,1:x_node-1],levels=levels)
plt.title('Contours of u velocity distribution at 1s (Re = 20)')
pylab.xlabel('x(m)')
pylab.ylabel('y(m)')
pylab.colorbar(CS)
plt.show()

plt.figure(32)
levels = np.linspace(-0.003,0.015,50)
CS = plt.contourf(x_grid_half, y_grid_half, Re_20_u_2s[1:y_node-1,1:x_node-1],levels=levels)
plt.title('Contours of u velocity distribution at 2s (Re = 20)')
pylab.xlabel('x(m)')
pylab.ylabel('y(m)')
pylab.colorbar(CS)
plt.show()

plt.figure(33)
levels = np.linspace(-0.003,0.015,50)
CS = plt.contourf(x_grid_half, y_grid_half, Re_20_u_4s[1:y_node-1,1:x_node-1],levels=levels)
plt.title('Contours of u velocity distribution at 4s (Re = 20)')
pylab.xlabel('x(m)')
pylab.ylabel('y(m)')
pylab.colorbar(CS)
plt.show()

plt.figure(34)
levels = np.linspace(-0.003,0.015,50)
CS = plt.contourf(x_grid_half, y_grid_half, Re_20_u_8s[1:y_node-1,1:x_node-1],levels=levels)
plt.title('Contours of u velocity distribution at 8s (Re = 20)')
pylab.xlabel('x(m)')
pylab.ylabel('y(m)')
pylab.colorbar(CS)

```

```

plt.show()

plt.figure(35)
levels = np.linspace(-0.003,0.015,50)
CS = plt.contourf(x_grid_half, y_grid_half, Re_20_u_steady[1:y_node-1,1:x_node-1],levels=levels)
plt.title('Contours of u velocity distribution at steady state (Re = 20)')
pylab.xlabel('x(m)')
pylab.ylabel('y(m)')
pylab.colorbar(CS)
plt.show()

plt.figure(41)
levels = np.linspace(np.amin(Re_20_v_1s),np.amax(Re_20_v_1s),50)
CS = plt.contourf(x_grid_half, y_grid_half, Re_20_v_1s[1:y_node-1,1:x_node-1],levels=levels)
plt.title('Contours of v velocity distribution at 1s (Re = 20)')
pylab.xlabel('x(m)')
pylab.ylabel('y(m)')
pylab.colorbar(CS)
plt.show()

plt.figure(42)
levels = np.linspace(np.amin(Re_20_v_2s),np.amax(Re_20_v_2s),50)
CS = plt.contourf(x_grid_half, y_grid_half, Re_20_v_2s[1:y_node-1,1:x_node-1],levels=levels)
plt.title('Contours of v velocity distribution at 2s (Re = 20)')
pylab.xlabel('x(m)')
pylab.ylabel('y(m)')
pylab.colorbar(CS)
plt.show()

plt.figure(43)
levels = np.linspace(np.amin(Re_20_v_4s),np.amax(Re_20_v_4s),50)
CS = plt.contourf(x_grid_half, y_grid_half, Re_20_v_4s[1:y_node-1,1:x_node-1],levels=levels)
plt.title('Contours of v velocity distribution at 4s (Re = 20)')
pylab.xlabel('x(m)')
pylab.ylabel('y(m)')
pylab.colorbar(CS)
plt.show()

plt.figure(44)
levels = np.linspace(np.amin(Re_20_v_8s),np.amax(Re_20_v_8s),50)
CS = plt.contourf(x_grid_half, y_grid_half, Re_20_v_8s[1:y_node-1,1:x_node-1],levels=levels)

```

```

plt.title('Contours of v velocity distribution at 8s (Re = 20)')
pylab.xlabel('x(m)')
pylab.ylabel('y(m)')
pylab.colorbar(CS)
plt.show()

plt.figure(45)
levels = np.linspace(np.amin(Re_20_v_steady),np.amax(Re_20_v_steady),50)
CS = plt.contourf(x_grid_half, y_grid_half, Re_20_v_steady[1:y_node-1,1:x_node-1],levels=levels)
plt.title('Contours of v velocity distribution at steady state (Re = 20)')
pylab.xlabel('x(m)')
pylab.ylabel('y(m)')
pylab.colorbar(CS)
plt.show()

plt.figure(51)
levels = np.linspace(-0.009,0.045,50)
CS = plt.contourf(x_grid_half, y_grid_half, Re_50_u_1s[1:y_node-1,1:x_node-1],levels=levels)
plt.title('Contours of u velocity distribution at 1s (Re = 50)')
pylab.xlabel('x(m)')
pylab.ylabel('y(m)')
pylab.colorbar(CS)
plt.show()

plt.figure(52)
levels = np.linspace(-0.009,0.045,50)
CS = plt.contourf(x_grid_half, y_grid_half, Re_50_u_2s[1:y_node-1,1:x_node-1],levels=levels)
plt.title('Contours of u velocity distribution at 2s (Re = 50)')
pylab.xlabel('x(m)')
pylab.ylabel('y(m)')
pylab.colorbar(CS)
plt.show()

plt.figure(53)
levels = np.linspace(-0.009,0.045,50)
CS = plt.contourf(x_grid_half, y_grid_half, Re_50_u_4s[1:y_node-1,1:x_node-1],levels=levels)
plt.title('Contours of u velocity distribution at 4s (Re = 50)')
pylab.xlabel('x(m)')

```

```
pylab.ylabel('y(m)')
pylab.colorbar(CS)
plt.show()
```

```
plt.figure(54)
levels = np.linspace(-0.009,0.045,50)
CS = plt.contourf(x_grid_half, y_grid_half, Re_50_u_8s[1:y_node-1,1:x_node-1],levels=levels)
plt.title('Contours of u velocity distribution at 8s (Re = 50)')
pylab.xlabel('x(m)')
pylab.ylabel('y(m)')
pylab.colorbar(CS)
plt.show()
```

```
plt.figure(55)
levels = np.linspace(-0.009,0.045,50)
CS = plt.contourf(x_grid_half, y_grid_half, Re_50_u_steady[1:y_node-1,1:x_node-1],levels=levels)
plt.title('Contours of u velocity distribution at steady state (Re = 50)')
pylab.xlabel('x(m)')
pylab.ylabel('y(m)')
pylab.colorbar(CS)
plt.show()
```

```
plt.figure(61)
levels = np.linspace(np.amin(Re_50_v_1s),np.amax(Re_50_v_1s),50)
CS = plt.contourf(x_grid_half, y_grid_half, Re_50_v_1s[1:y_node-1,1:x_node-1],levels=levels)
plt.title('Contours of v velocity distribution at 1s (Re = 50)')
pylab.xlabel('x(m)')
pylab.ylabel('y(m)')
pylab.colorbar(CS)
plt.show()
```

```
plt.figure(62)
levels = np.linspace(np.amin(Re_50_v_2s),np.amax(Re_50_v_2s),50)
CS = plt.contourf(x_grid_half, y_grid_half, Re_50_v_2s[1:y_node-1,1:x_node-1],levels=levels)
plt.title('Contours of v velocity distribution at 2s (Re = 50)')
pylab.xlabel('x(m)')
pylab.ylabel('y(m)')
pylab.colorbar(CS)
plt.show()
```

```

plt.figure(63)
levels = np.linspace(np.amin(Re_50_v_4s),np.amax(Re_50_v_4s),50)
CS = plt.contourf(x_grid_half, y_grid_half, Re_50_v_4s[1:y_node-1,1:x_node-1],levels=levels)
plt.title('Contours of v velocity distribution at 4s (Re = 50)')
pylab.xlabel('x(m)')
pylab.ylabel('y(m)')
pylab.colorbar(CS)
plt.show()

```

```

plt.figure(64)
levels = np.linspace(np.amin(Re_50_v_8s),np.amax(Re_50_v_8s),50)
CS = plt.contourf(x_grid_half, y_grid_half, Re_50_v_8s[1:y_node-1,1:x_node-1],levels=levels)
plt.title('Contours of v velocity distribution at 8s (Re = 50)')
pylab.xlabel('x(m)')
pylab.ylabel('y(m)')
pylab.colorbar(CS)
plt.show()

```

```

plt.figure(65)
levels = np.linspace(np.amin(Re_50_v_steady),np.amax(Re_50_v_steady),50)
CS = plt.contourf(x_grid_half, y_grid_half, Re_50_v_steady[1:y_node-1,1:x_node-1],levels=levels)
plt.title('Contours of v velocity distribution at steady state (Re = 50)')
pylab.xlabel('x(m)')
pylab.ylabel('y(m)')
pylab.colorbar(CS)
plt.show()

```

```

plt.figure(101)
plt.plot(y_grid_half,Re_20_u_steady[1:y_node-1,20],y_grid_half,Re_50_u_steady[1:y_node-1,20],y_grid_half,Re_100_u_steady[1:y_node-1,20])
plt.title('u-velocity profile along vertical centerline at steady state')
pylab.xlabel('y(m)')
pylab.ylabel('u(m/s)')
plt.legend(['Re = 20', 'Re = 50', 'Re = 100'], loc='upper left')
plt.show()

```

```

plt.figure(102)
plt.plot(x_grid_half,Re_20_v_steady[20,1:x_node-1],x_grid_half,Re_50_v_steady[20,1:x_node-1],x_grid_half,Re_100_v_steady[20,1:x_node-1])
plt.title('v-velocity profile along horizontal centerline at steady state')

```

```

pylab.xlabel('x(m)')
pylab.ylabel('v(m/s)')
plt.legend(['Re = 20', 'Re = 50', 'Re = 100'], loc='upper left')
plt.show()

color=np.sqrt(np.multiply(Re_100_u_steady[1:y_node-1,1:x_node-
1],Re_100_u_steady[1:y_node-1,1:x_node-1])+np.multiply(Re_100_v_steady[1:y_node-
1,1:x_node-1],Re_100_v_steady[1:y_node-1,1:x_node-1]))

plt.figure(1)
SP = plt.streamplot(x_grid_half, y_grid_half, Re_100_u_steady[1:y_node-1,1:x_node-
1],Re_100_v_steady[1:y_node-1,1:x_node-1],density=3,color=color,linewidth=1)
plt.title('Streamline of velocity at steady state (Re = 100)')
pylab.xlabel('x(m)')
pylab.ylabel('y(m)')
cb = pylab.colorbar(SP.lines)
plt.show()

color=np.sqrt(np.multiply(Re_50_u_steady[1:y_node-1,1:x_node-
1],Re_50_u_steady[1:y_node-1,1:x_node-1])+np.multiply(Re_50_v_steady[1:y_node-
1,1:x_node-1],Re_50_v_steady[1:y_node-1,1:x_node-1]))

plt.figure(2)
SP = plt.streamplot(x_grid_half, y_grid_half, Re_50_u_steady[1:y_node-1,1:x_node-
1],Re_50_v_steady[1:y_node-1,1:x_node-1],density=3,color=color,linewidth=1)
plt.title('Streamline of velocity at steady state (Re = 50)')
pylab.xlabel('x(m)')
pylab.ylabel('y(m)')
pylab.colorbar(SP.lines)
plt.show()

color=np.sqrt(np.multiply(Re_20_u_steady[1:y_node-1,1:x_node-
1],Re_20_u_steady[1:y_node-1,1:x_node-1])+np.multiply(Re_20_v_steady[1:y_node-
1,1:x_node-1],Re_20_v_steady[1:y_node-1,1:x_node-1]))

plt.figure(3)
SP = plt.streamplot(x_grid_half, y_grid_half, Re_20_u_steady[1:y_node-1,1:x_node-
1],Re_20_v_steady[1:y_node-1,1:x_node-1],density=3,color=color,linewidth=1)
plt.title('Streamline of velocity at steady state (Re = 20)')
pylab.xlabel('x(m)')
pylab.ylabel('y(m)')
pylab.colorbar(SP.lines)

```

```
plt.show()
```