COMPUTER ASSIGNMENT 3 REPORT

Pratham Sunkad

ME21B145

IIT Madras

AM 5630 Foundations of CFD

Prof. Arul Prakash

1st May 2024

1. Problem Definition:

A square cavity of side, a=1 unit as shown in Figure 1. The cavity is filled with an incompressible fluid. The flow is steady and 2-Dimensional.

Determine the fluid flow pattern and pressure distribution inside the cavity at Reynolds number, Re = 100 using **Stream function vorticity method** or **SMAC** method or any other finite difference method learnt in this course. Use the non-dimensional form of the governing equation with appropriate boundary conditions.

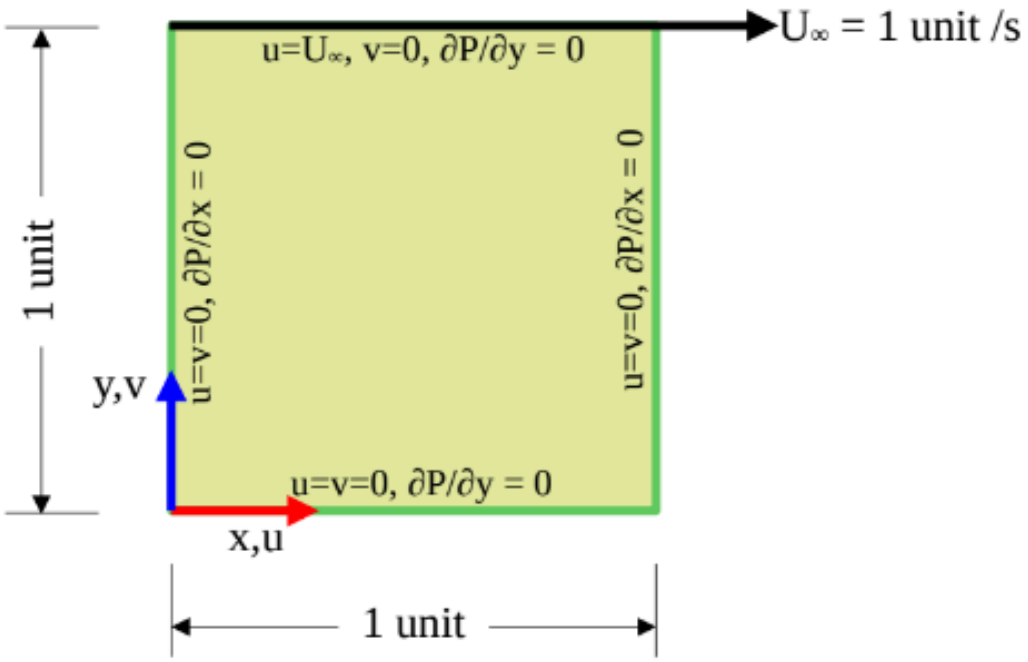


Figure 1: Computational domain of Lid-driven square cavity

I am using the **Stream Function Vorticity Approach**

2. Governing Equation:

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

Vorticity relation with velocity

Stream function relation with v, u

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

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

New Momentum equation by eliminating u,v and substituting with vorticity

$$\frac{\partial \zeta}{\partial t} + u\frac{\partial \zeta}{\partial x} + v\frac{\partial \zeta}{\partial y} = \nu\left(\frac{\partial^2 \zeta}{\partial x^2} + \frac{\partial^2 \zeta}{\partial y^2}\right)$$

Relation between stream function and vorticity

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

Pressure relation

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

3. Initial and Boundary Conditions:

Initial Conditions:

Top wall : u = 1, v = 0

Left wall : u = 0, v = 0

Right wall : u = 0, v = 0

Bottom wall : u = 0, v = 0

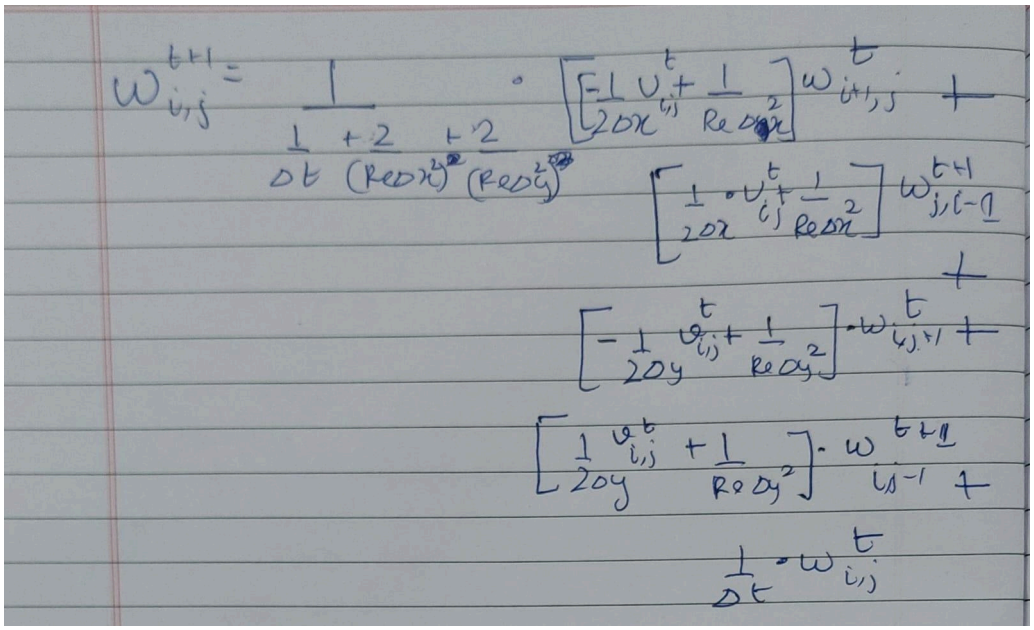
Boundary Conditions:

$\partial P/\partial y = 0$ at bottom and top wall

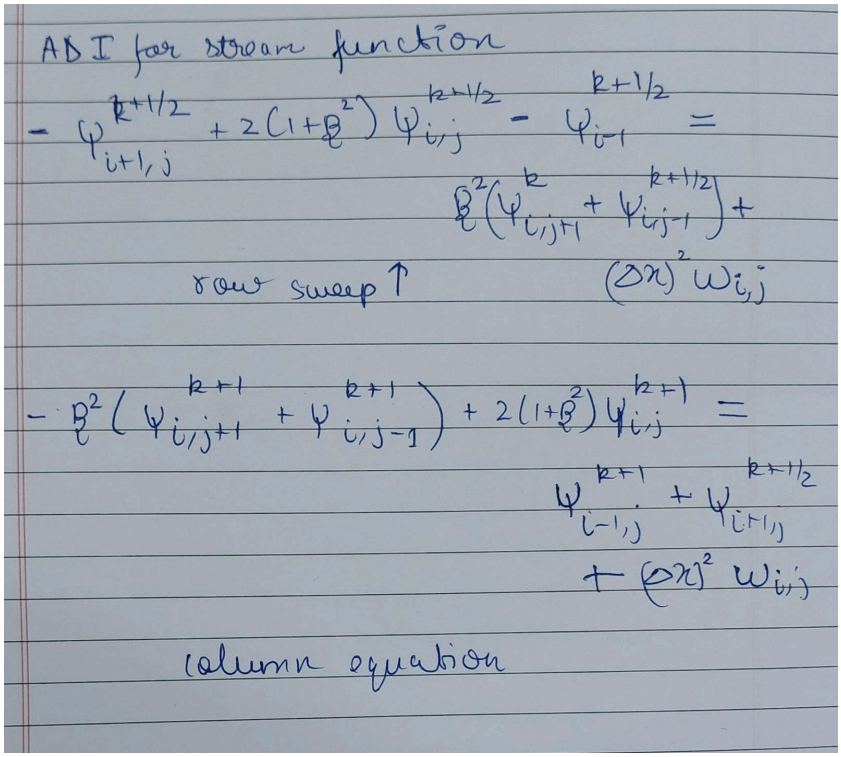
$\partial P/\partial x = 0$ at right and left wall

4. Numerical Formulation:

1) **Point Gauss Seidel** :. Point Gauss Seidel is used to solve the **Vorticity transport equation**. This Equation is solved at every time step to obtain updated value of **vorticity**



2) **ADI** : Introduce $\psi^{k+½}$ by sweeping over rows first. Use $\psi^{k+½}$ to calculate ψ^{k+1} by sweeping over the columns. Solve the Tridiagonal matrix at each row/column sweep to get stream function values. This is used to solve for **stream function values at each time step after obtaining values of vorticity.**



3) **ADI for Pressure:** After calculating the **steady values of vorticity** and **stream function** we can go ahead and calculate the **pressure distribution**. ADI method is used for this and the formulation is similar to stream function.

$$\nabla^2 p = 2\rho_{i,j}\left[\left(\frac{\psi_{i+1,j} - 2\psi_{i,j} + \psi_{i-1,j}}{(\Delta x)^2}\right)\left(\frac{\psi_{i,j+1} - 2\psi_{i,j} + \psi_{i,j-1}}{(\Delta y)^2}\right) - \left(\frac{\psi_{i+1,j+1} - \psi_{i+1,j-1} - \psi_{i-1,j+1} + \psi_{i-1,j-1}}{4\,\Delta x\,\Delta y}\right)^2\right]$$

This equation is discretized using ADI. Introduce $\mathbf{P}^{k+\frac{1}{2}}$ by sweeping over rows first. Use $\mathbf{P}^{k+\frac{1}{2}}$ to calculate $\mathbf{P}^{k+1}$ by sweeping over the columns. Solve the Tridiagonal matrix at each row/column sweep to get Pressure values..

5. PseudoCode:

a) Procedure :

1. Specify initial values for ζ and ψ at time $t = 0$.
2. Solve the vorticity transport equation for ζ at each interior grid point at time $t + \Delta t$.
3. Iterate for new ψ values at all points by solving the Poisson equation using new ζ at interior points.
4. Find the velocity components from $u = \psi_y$ and $v = -\psi_x$.
5. Determine values of ζ on the boundaries using ψ and ζ values at interior points.
6. Return to Step 2 if the solution is not converged.

Step 1 : Initializing vorticity and stream function:

1: Declare all variables such as Δx , Δy , β , Re.

2 : Initialize the **u & v** velocity at each point given the initial conditions.

3: Use the **vorticity** and **velocity** relation to find the **vorticity** at the interior points. For the boundary points use the Image point technique to find vorticity values.

4: Calculate the stream function using the vorticity relation.

5: Use this stream function to find vorticity values at the boundary using Image point technique

Pseudo Code:

```
for i = 1: n_y
    u_y(i,:) = (u_array(i+1,:) - u_array(i-1,:))/(2*del_y);
For j = 1: n_x
    v_x(:,j) = (v_array(:,j+1) - v_array(:, j-1))/(2*del_x);
vor_array = v_x - u_y;


[stream_func, iteration] = cal_streamfunc(vor_array, Lx, Ly);
vor_array(:, 1) = -2*stream_func(:,2)/(del_x^2);
vor_array(:,end) = -2*stream_func(:,end-1)/(del_x^2);
vor_array(1,:) = -2*stream_func(2,:)/(del_y^2);
vor_array(end,:) = -2*(stream_func(end-1,:) + del_y*u0)/(del_y^2);
```

Step 2 : Solving the Vorticity Transport Equation

1: Declare all variables such as Δx , Δy , β , Re.

2: Solve the Vorticity transport equation at time step ‘t’ using the Gauss Seidel Formulation.

3: Create two 2D matrices vor_prev and new_vor_array for storing the value of temperature at k and $k + 1$ iteration of size N × M where, M is number of grid points in x-direction and N is number of grid points in y-direction..

4 : Pseudo Code:

```
while true
   vor_prev = new_vor_array;
   for i = 2:n_x - 1
```

```
      for j = 2:n_y-1

          new_vor_array(j,i) = (1/factor)*(u_mod_1(j,i)*vor_prev(j,i+1) +
u_mod_2(j,i)*new_vor_array(j,i-1) + v_mod_1(j,i)*vor_prev(j+1,i) +
v_mod_2(j,i)*new_vor_array(j-1,i) + (1/del_t)*vor_array(j,i));
```

5 : Solve the Gauss Seidel until convergence to obtain values of vorticity.

Step 3 : Solving for stream function

1: Declare all variables such as Δx , Δy , β , Re.

2: Use the vorticity calculated in the previous step to calculate stream function at each. A poisson equation relates both of them.

3: Formulate the ADI method for solving this equation.

Create two 2D matrices stream_half and stream_func for storing the value of temperature at k +1/2 and k iteration of size N × M where, M is number of grid points in x-direction and N is number of grid points in y-direction.

Create stream_check which is of size N × M with all interior elements 0 and the boundary conditions applied. stream_check will have elements at $k + 1$ iteration at the end of a for loop.

4: Pseudo Code

```
stream_func = zeros(n_y, n_x);

iteration = 1;

stream_bound = stream_func;

tolerance = 0.02;

while true

   stream_check = stream_bound;

   stream_half = stream_bound;

   %row sweep
```

```
   for j = 2: n_y-1
       rhs = transpose(beta*stream_func(j+1, 2: n_x-1) + beta*stream_half(j-1,
2: n_x-1) + ((del_x)^2)*vor_array(j,2:n_x-1));
       tri_diag_mat = zeros((n_x-2),(n_x-2));
       for i = 2:n_x-1
              if i == 2
                  tri_diag_mat(i-1,i-1) = 2*(1+beta);
                  tri_diag_mat(i,i-1) = -1;
                  rhs(i-1) = rhs(i-1) + stream_bound(j, i-1);
                  continue
              end
              if i == n_x - 1
                  tri_diag_mat(i-1,i-1) = 2*(1+beta);
                  tri_diag_mat(i-2,i-1) = -1;
                  rhs(i-1) = rhs(i-1) + stream_bound(j, i+1);
                  continue
              end
              tri_diag_mat(i-1,i-1) = 2*(1+beta);
              tri_diag_mat(i,i-1) = -1;
              tri_diag_mat(i-2,i-1) = -1;
       end
       temp_array = tdma(tri_diag_mat, rhs);
       stream_half(j,2: n_x - 1) = transpose(temp_array);
   end
```

5: Similarly for the column sweep. Solve the matrix using the TDMA algorithm.

Step 4: Obtaining new velocity values

1: Declare all variables such as Δx , Δy , β , Re

2 : After obtaining the stream function values, use the relation **between stream function and velocity** to find values of **u & v** at each point.

3 : Pseudo Code

```
[u,v] = initialize_u_v(n_x, n_y, u0);
for i=2:n_y-1
   u(i,:) = (stream_func(i+1,:) - stream_func(i-1,:))/(2*del_y);
for i=2: n_x-1
   v(:,i) = -1*(stream_func(:,i+1) - stream_func(:,i-1))/(2*del_x);
```

4 : Use new velocity values in the next iteration.

If Not Converged → Step 2

Else → Terminate and Calculate pressure distribution

Step 5 : Calculate Pressure

Step 1: Declare all variables such as Δx , Δy , β , Re.

2: Calculate Matrices $(d^2\boldsymbol{\psi}/d^2x) * (d^2\boldsymbol{\psi}/d^2y)$ and $(d^2\boldsymbol{\psi}/dy\,dx))^2$ to formulate the ADI method

```
stream_grad_xx = zeros(n_y, n_x);
stream_grad_yy = zeros(n_y, n_x);
stream_grad_x_y = zeros(n_y, n_x);
for i = 2: n_x - 1
   for j = 2: n_y - 1
       stream_grad_yy(j,i) = (stream_func(j+1,i) - 2*stream_func(j,i) +
stream_func(j-1,i))/(del_y^2);
```

```
       stream_grad_xx(j,i) = (stream_func(j,i+1) - 2*stream_func(j,i) +
stream_func(j,i-1))/(del_x^2);

       stream_grad_x_y(j,i) = (((stream_func(j+1,i+1) - stream_func(j-1,i+1) -
stream_func(j+1,i-1) + stream_func(j-1,i-1))/(4*del_x*del_y)))^2;

   end

end

stream_net = stream_grad_xx.*stream_grad_yy;

beta = (del_x/del_y)^2;

pressure_array = zeros(n_y, n_x);
```

3: Formulate the Tridiagonal Matrix for row and column sweep and solve it using the Tridiagonal Matrix algorithm until convergence.

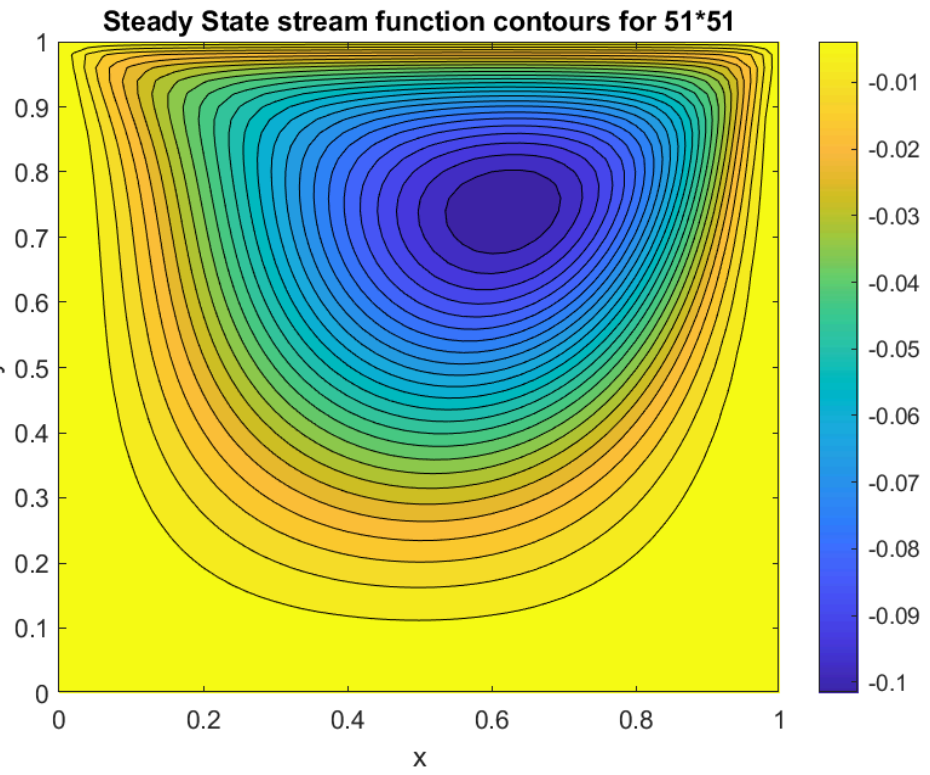
4 : Plot the pressure distribution

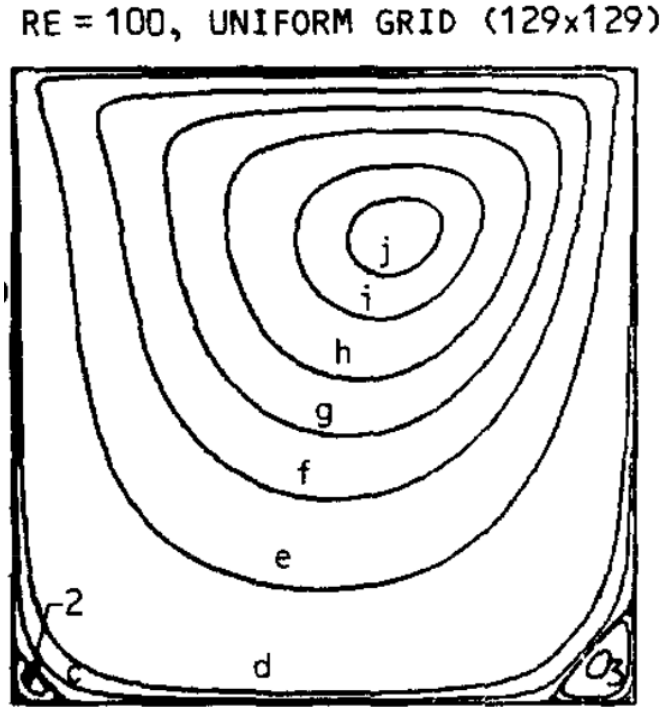
TABLE:

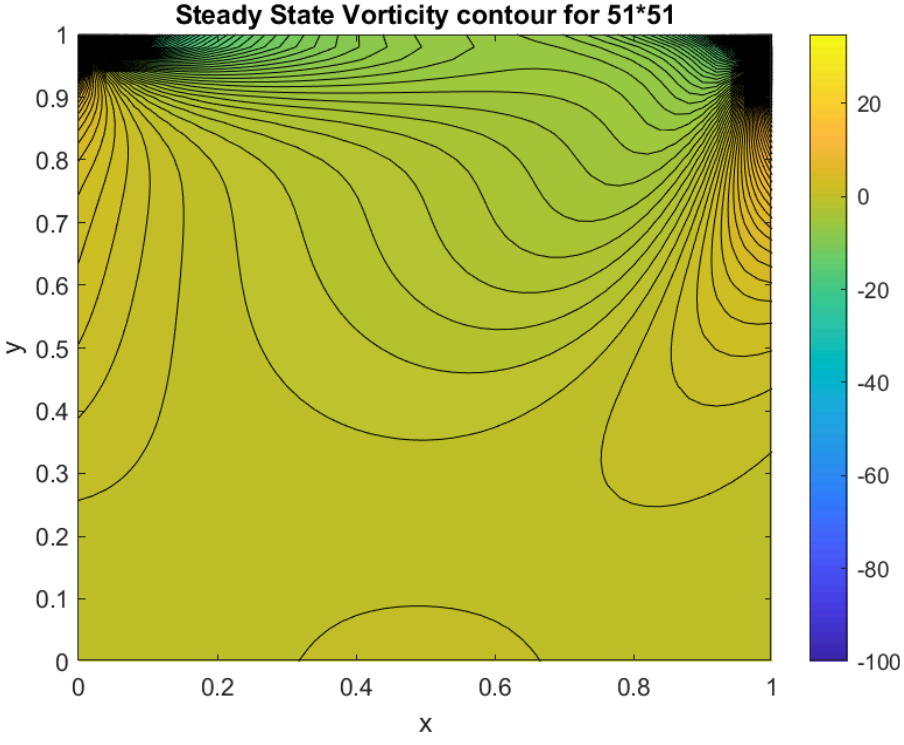
S No.	Scheme Used	Grid Size	Δt	Re	Convergence Criteria for all methods:	No. of Iterations
1	Stream function Vorticity	51 x 51	0.01	100	$\Sigma \lvert u^{k+1} - u^{k} \rvert < 0.02$	878
2	Stream function Vorticity	101 x 101	0.01	100	$\Sigma \lvert u^{k+1} - u^{k} \rvert < 0.02$	1143

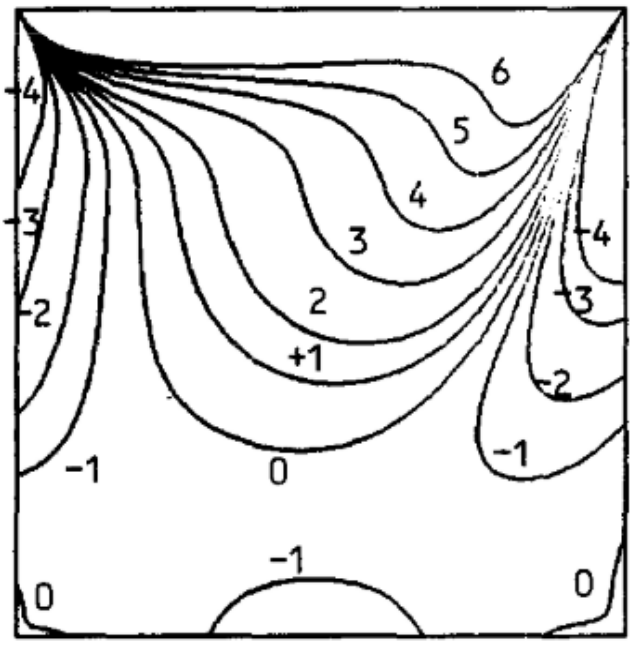
6. Results:

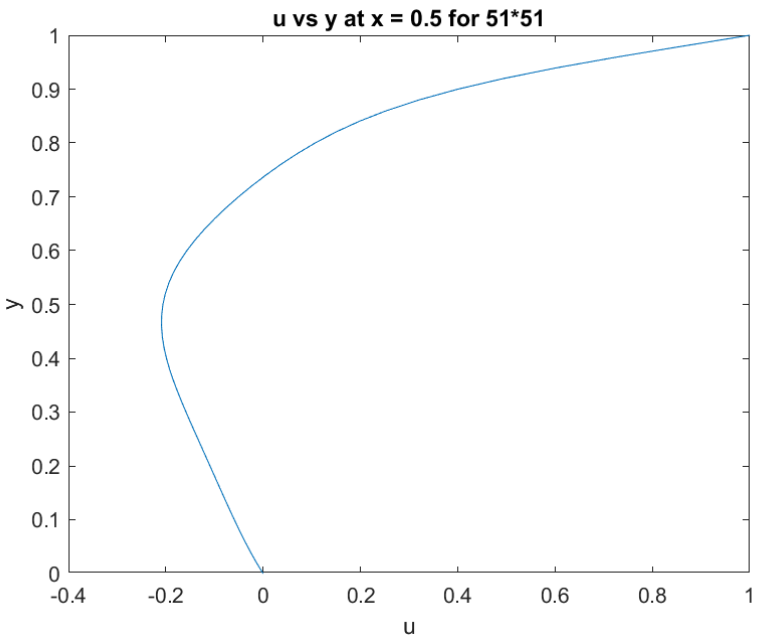
For Grid Size 51 *51

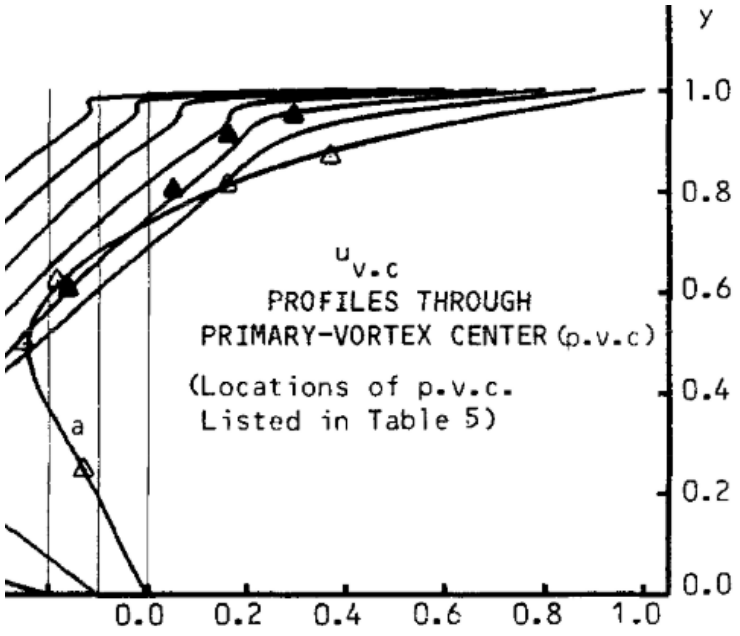


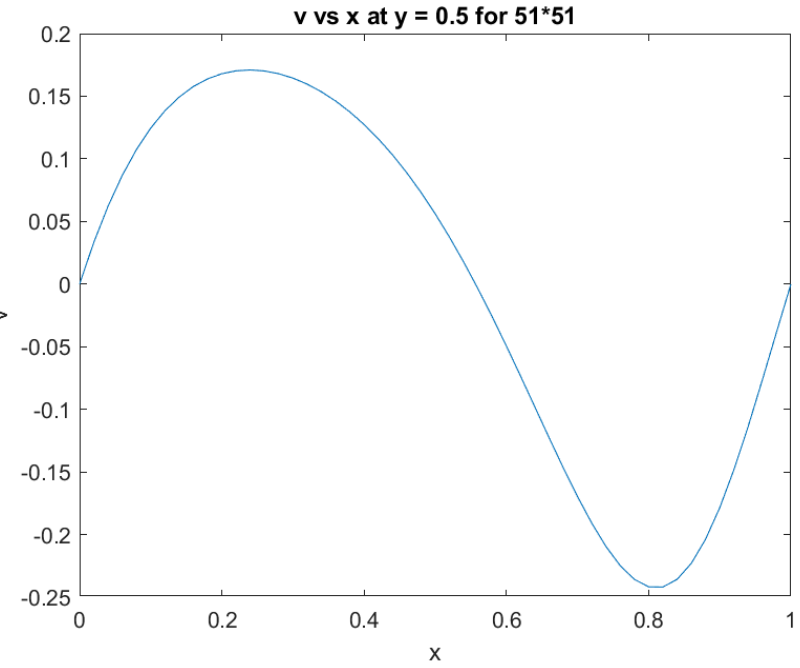


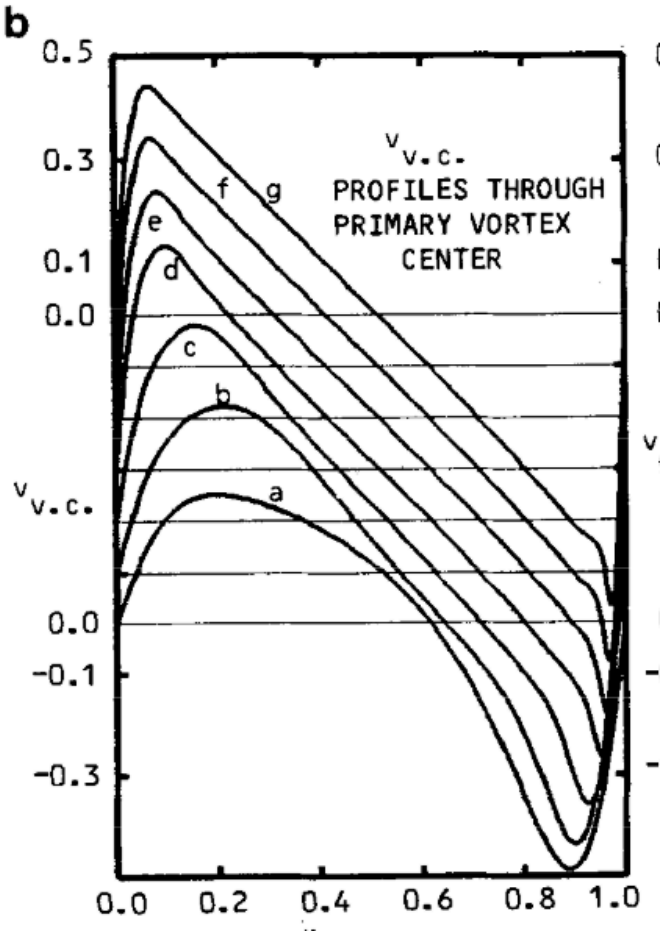


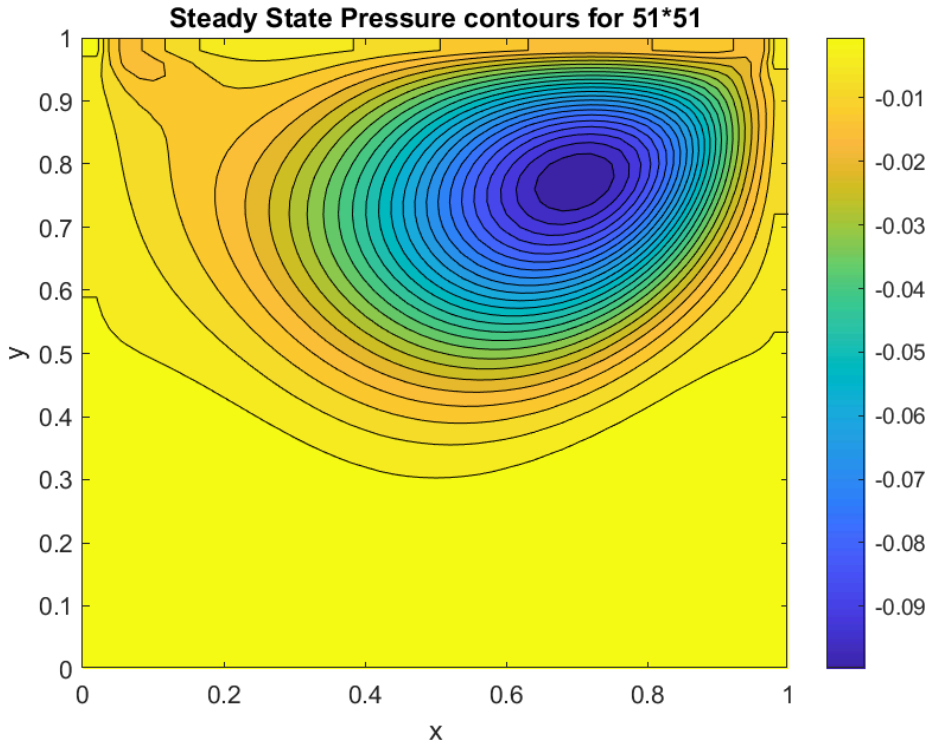



u vs y at x = 0.5 for 51*51
y
u
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

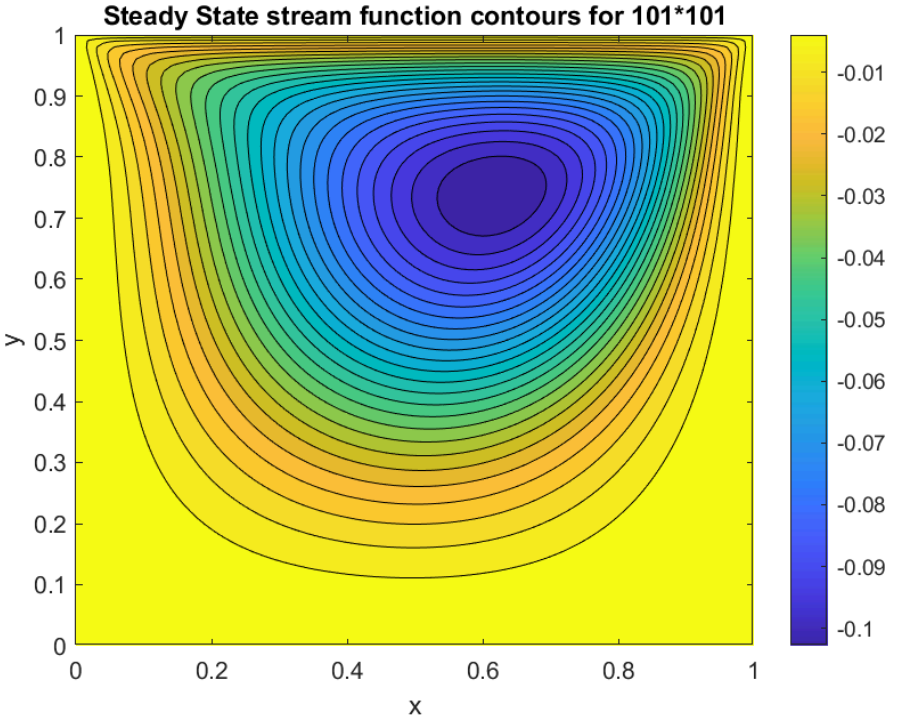

y
1.0
0.8
0.6
0.4
0.2
0.0
u v.c
PROFILES THROUGH
PRIMARY-VORTEX CENTER (p.v.c)
(Locations of p.v.c.
Listed in Table 5)
a
0.0
0.2
0.4
0.6
0.8
1.0

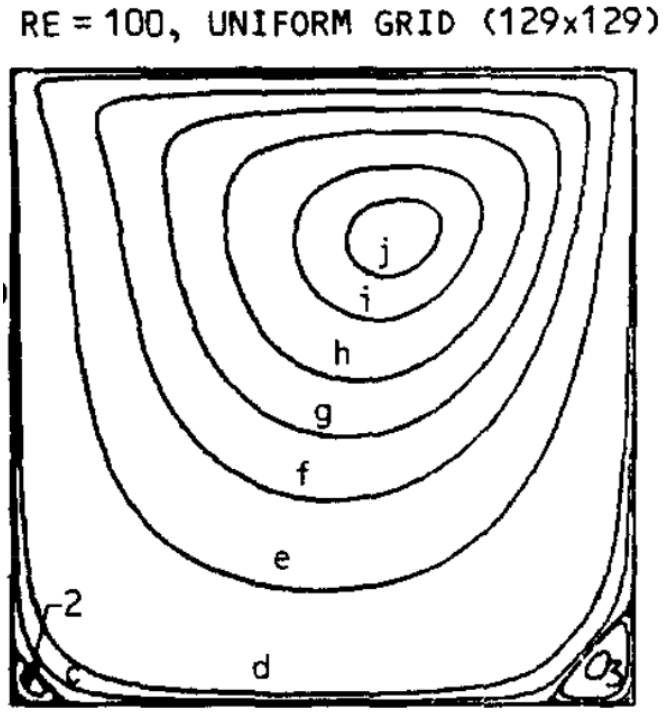

v vs x at y = 0.5 for 51*51
v
x
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
0
0.2
0.4
0.6
0.8
1

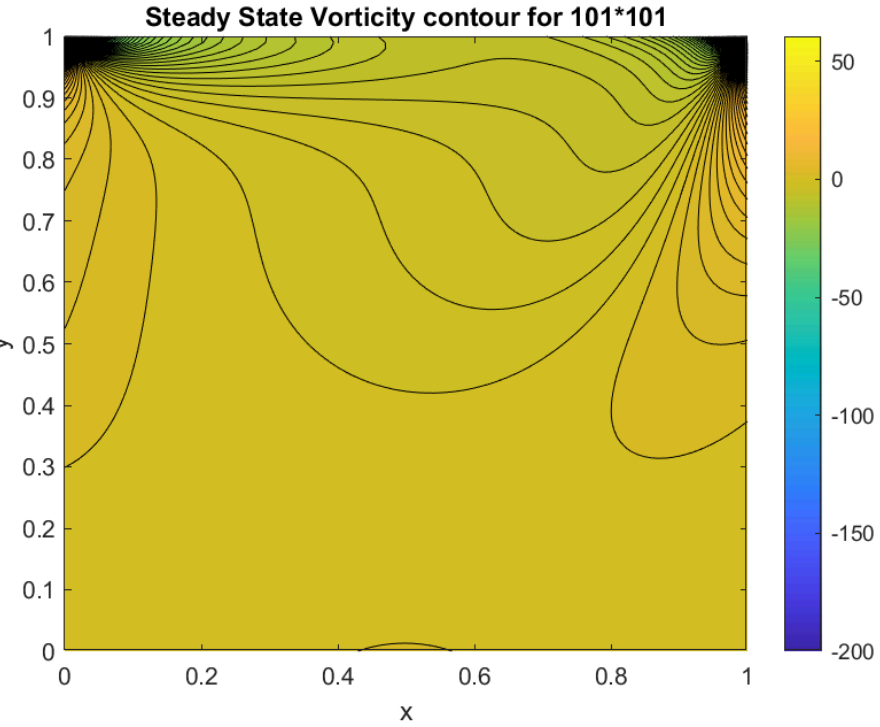

b
0.5
0.3
0.1
0.0
-0.1
-0.3
v v.c.
PROFILES THROUGH
PRIMARY VORTEX
CENTER
f
g
e
d
c
b
a
v v.c.
0.0
0.2
0.4
0.6
0.8
1.0

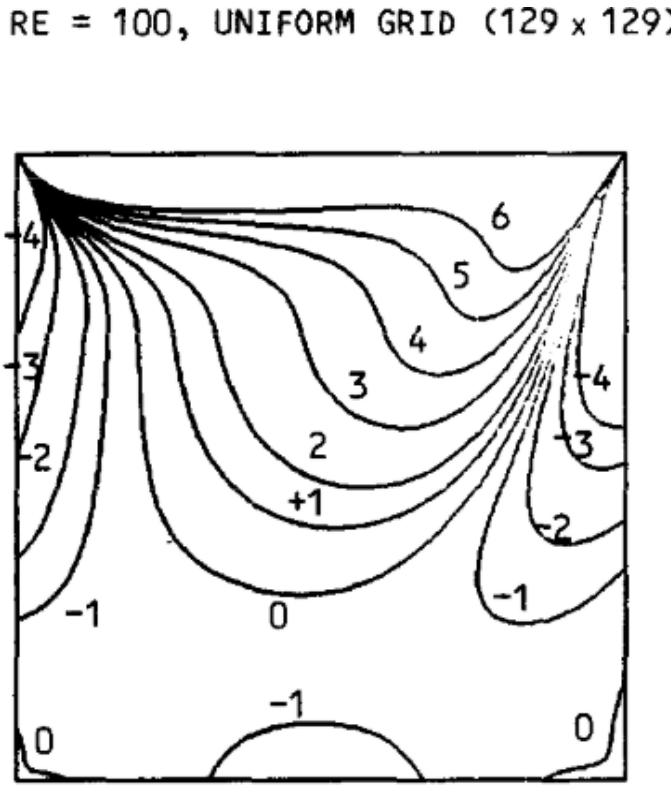


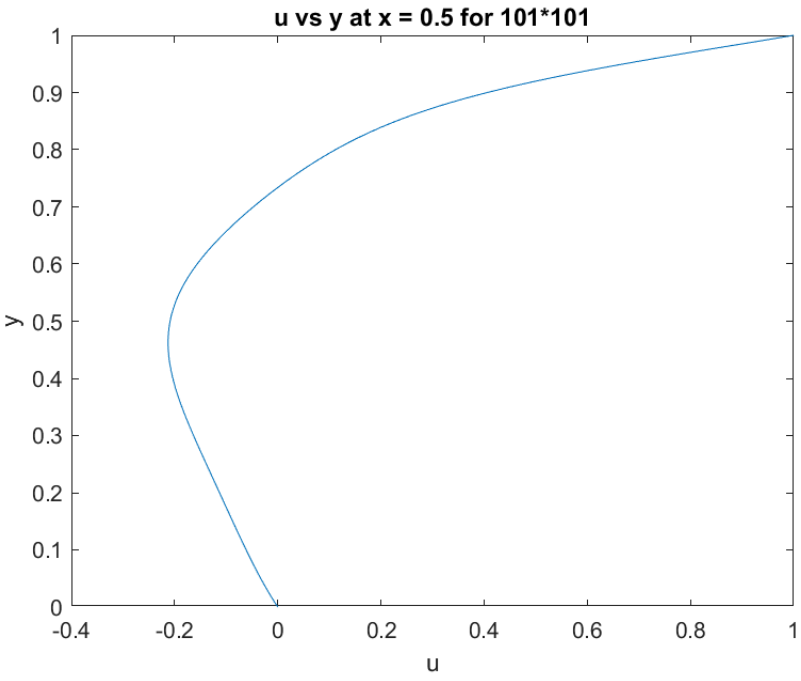
For Grid Size 101 *101

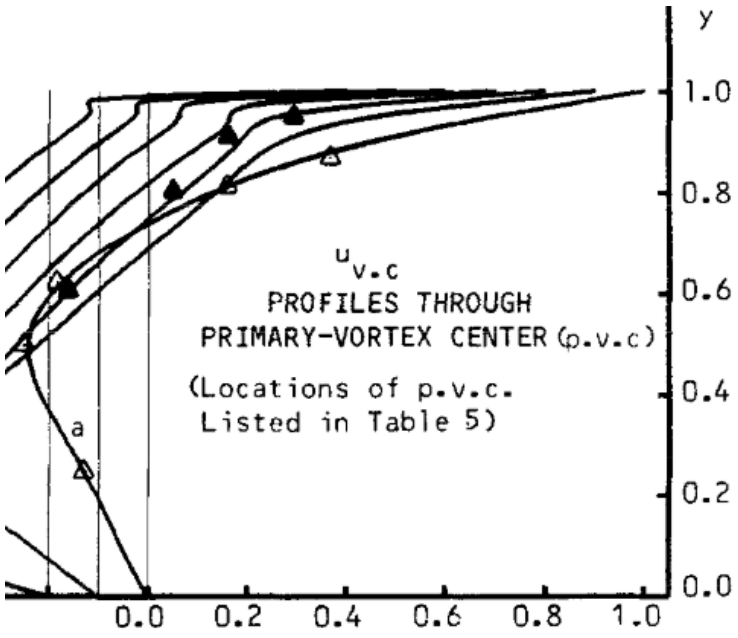


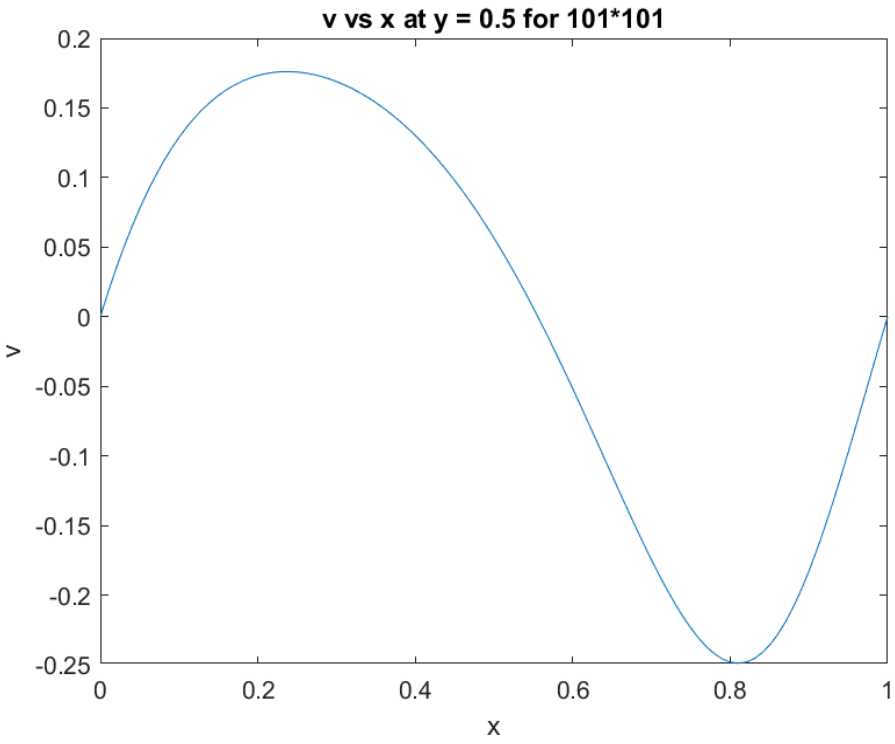


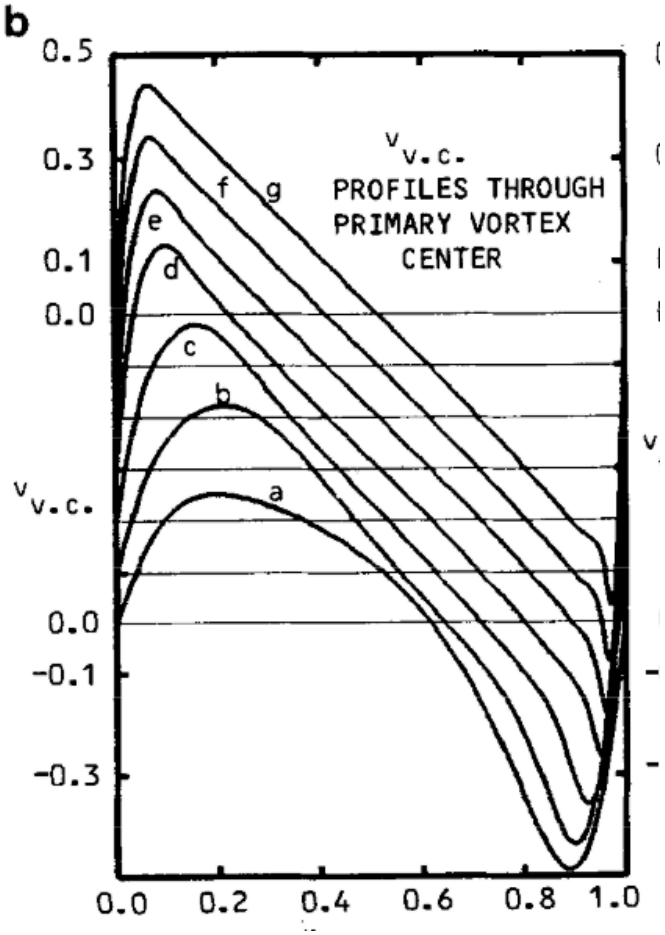

Steady State Vorticity contour for 101*101
x
y
50
0
-50
-100
-150
-200

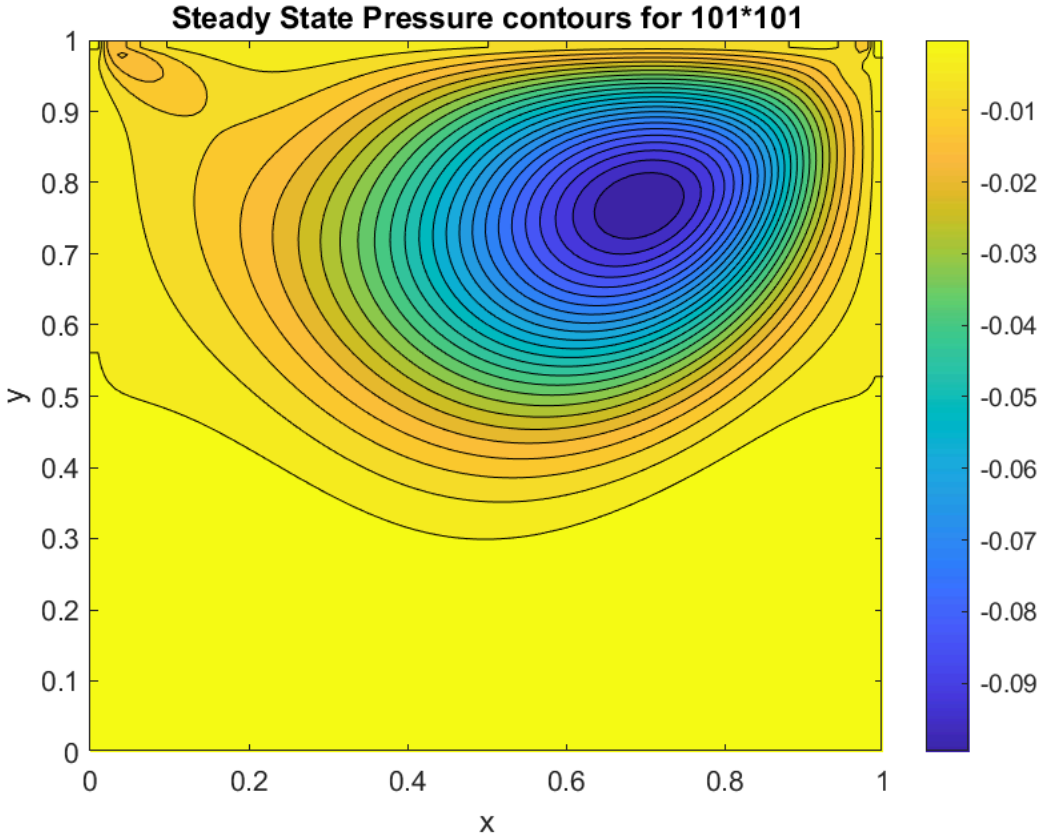

RE = 100, UNIFORM GRID (129 x 129)
6
5
4
3
2
+1
-1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
-1
0


u vs y at x = 0.5 for 101*101
u
y


$u_{v.c}$
PROFILES THROUGH
PRIMARY-VORTEX CENTER (p.v.c)
(Locations of p.v.c.
Listed in Table 5)
a
y


v vs x at y = 0.5 for 101*101
v
x
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
0
0.2
0.4
0.6
0.8
1


b
0.5
0.3
0.1
0.0
-0.3
$v_{v.c.}$
PROFILES THROUGH
PRIMARY VORTEX
CENTER
a
b
c
d
e
f
g
0.0
0.2
0.4
0.6
0.8
1.0


Steady State Pressure contours for 101*101
y
x
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.2
0.4
0.6
0.8
1
-0.01
-0.02
-0.03
-0.04
-0.05
-0.06
-0.07
-0.08
-0.09

Discussion:

1) Grid Size 51*51 converged in 878 iterations
2) Grid Size 101*101 converged in 1143 iterations
3) The plots from my experiment match that of the paper
4) Point Gauss seidel was used to solve vorticity equation
5) ADI method was used to solve for stream-function and pressure

Computer Code:

S No.	Task	Name of the Matlab File
1	Initialize vorticity & stream_func	initial_vorticity.mlx
2	Calculate stream_func using vorticity	cal_streamfunc.mlx
3	Performs ADI to find stream_func	ADI_streamfunc.mlx
4	Initializes u and v for each point	initialize_u_v.mlx
5	Calculates u & v using stream_func	cal_u_v.mlx
6	Performs the entire algorithm	vorticity_eqn.mlx
7	Gauss Seidel method to find vorticity	Point_GS_vorticity.mlx
8	Tridiagonal Matrix Algorithm	tdma.mlx
9	Calculates Pressure using ADI	cal_
9	Running all the codes	Main_A3.mlx

—----------end—----------