MAE 561
Computational Fluid Dynamics
Final Project

Simulation of Lid Driven Cavity Problem using Incompressible Navier-Strokes Equation

AKSHAY BATRA
1205089388

TABLE OF CONTENTS

1. Abstract……………………………………………………………………………………………....3

2. Acknowledgement…………………………………………………………………………...4

3. Introduction………………………………………………………………………………….5

4. Problem Statement ……………………………………………………………………….5

5. Governing Equations…………………………………………..5
   5.1. Stream Function………………………………………………………………………...5
   5.2 Vorticity ……………………………………………………...6
      5.2.1 Boundary conditions for the vorticity………………………………………………6
   5.3 Stream function equation………………………………..7
   5.4 Successive Over relaxation…………………………………………………………… 7
   5.5 Pressure Calculation……………………………………..7

6. Development of the coding algorithm………………………………………………………8

7. Results……………………………………………………………………………………….9
   7.1. Numerical simulation results ……………………………….. 9
   7.2. Ansys Fluent results……………………………………………………………………18

8. Conclusions…………………...…………………………………………………………….20

9. References…………...……………………………………………………………….. 21

10. Appendix (matlab code)……………………………………..22

1. ABSTRACT

Develop an understanding of the steps involved in solving the Navier-Stokes equations using a numerical method. Write a simple code to solve the “driven cavity” problem using the Incompressible- Navier-Stokes equations in Vorticity form. This project requires that the Vorticity streamline function, u and v velocity profiles, pressure contours for the lid driven rectangular cavity for Reynolds number 100 and 1000. The lid driven cavity is a classical problem and closely resembles actual engineering problems that exist in research and industry areas. The vorticity equation will be solved utilizing a forward time central space (FTCS) explicit method. The streamline equation is solved using the successive over relaxation method. The obtained results follows and are illustrated in the report.

2. ACKNOWLEDGEMENTS

I would take this opportunity to thank Dr. H.P Huang, my instructor for this course, Computational Fluid Dynamics (MAE 561). He has been a great help for me during this course. Without his support I wouldn’t have been able to achieve what I have achieved in this course. He has been very instrumental in my understanding of numerical methods for computational fluid dynamics.

Also I would like to thank Donley Hurd for his constant technical support during this course.

3. INTRODUCTION

In previous homework assignments an analysis of a how to solve partial differential equations (PDEs) using point Gauss-Seidel (PGS) iterative method and using forward time center space (FTCS) explicit method has been explored. In this project an analysis will be conducted that will utilize these two methods in one problem. But successive over relaxation (SOR) method would be used as the iteration method. This project will consider a rectangular cavity with a moving top wall. This moving wall will slowly cause the fluid to move within the cavity. It is the final steady state solution that this project seeks to acquire (Re 100 and 1000). Finally the similar problem is computed in ANSYS FLUENT, commercial fluid simulation software and results are compared.

4. PROBLEM STATEMENT

The upper plate of a rectangular cavity shown in Figure 1 moves to the rights with a velocity of u_o. The rectangular cavity has dimensions of L by L. Use the FTCS explicit scheme and the SOR formation to solve for the vorticity and the stream function equations, respectfully. The cavity flow problem is to be solved for the vorticity, streamline, pressure contours and u-v profiles for Re=100 and Re 1000. Later, a case has to be solved where the rectangular cavity as dimensions 2L and L to obtain same contours.

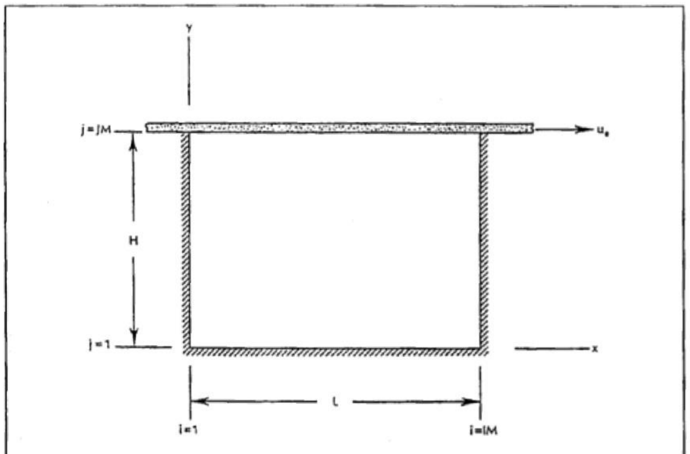


Figure P8-1. The domain of solution for the driven cavity problem.

Figure 1: (Hoffmann) Figure P8-1 page 357

5. GOVERNING EQUATIONS

5.1 Stream Function

The derivation for the FTCS starts with the vorticity equation seen in Equation 1. It is important to notice how similar this equation is to the 2-D Navier-Stokes momentum equation.

$$\frac{\partial \Omega}{\partial t} + u\frac{\partial \Omega}{\partial x} + v\frac{\partial \Omega}{\partial y} = \upsilon\left(\frac{\partial^2 \Omega}{\partial x^2} + \frac{\partial^2 \Omega}{\partial y^2}\right) \quad \textbf{(1)}$$

Equation 1 then has a forward difference Taylor Series expansion for first derivatives applied to the first term, a central difference Taylor Series expansion for first derivatives applied to the second term and third term, and central difference Taylor Series for second derivative applied to the fourth and fifth term. The result is shown in Equation 2.

$$\frac{\Omega_{i,j}^{n+1}-\Omega_{i,j}^{n}}{\Delta t}+u_{i,j}^{n}\frac{\Omega_{i+1,j}^{n}-\Omega_{i-1,j}^{n}}{2\Delta x}+v_{i,j}^{n}\frac{\Omega_{i,j+1}^{n}-\Omega_{i,j-1}^{n}}{2\Delta x}=\upsilon\left(\frac{\Omega_{i+1,j}^{n}-2\Omega_{i,j}^{n}+\Omega_{i-1,j}^{n}}{(\Delta x)^2}+\frac{\Omega_{i,j+1}^{n}-2\Omega_{i,j}^{n}+\Omega_{i,j-1}^{n}}{(\Delta y)^2}\right) \quad \textbf{(2)}$$

Also if, in equation 1,

$$u=\frac{\partial \Psi}{\partial y} \quad , \quad v=-\frac{\partial \Psi}{\partial x} \quad \textbf{(3)}$$

Then, the values of u and v are substituted in equation 1 to obtain equation 3,

$$\frac{\partial \Omega}{\partial t} = - \frac{\partial \Psi}{\partial y}\frac{\partial \Omega}{\partial x}+\frac{\partial \Psi}{\partial x}\frac{\partial \Omega}{\partial y}+\frac{1}{Re}\{\frac{\partial}{\partial x}\left(\frac{\partial \Omega}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{\partial \Omega}{\partial y}\right)\} \quad \textbf{(4)}$$

For this problem we are considering, dx=dy=ds, thus the final discretized equation becomes,

$$\Omega_{i,j,n+1} = \Omega_{i,j,n} - dt\frac{[(\Psi_{i,j+1,n} - \Psi_{i,j-1,n})\ (\Omega_{i+1,j,n} - \Omega_{i-1,j,n})]}{4ds*ds} - dt\frac{[(\Psi_{i+1,j,n} - \Psi_{i-1,j,n})\ (\Omega_{i,j+1,n} - \Omega_{i,j-1,n})]}{4ds*ds} + dt\frac{[(\Omega_{i+1,j,n} - \Omega_{i-1,j,n} - \Omega_{i,j+1,n} - \Omega_{i,j-1,n} - 4\,\Omega_{i,j,n})]}{Re*\ ds*ds} \quad \textbf{(5)}$$

Where dt is the time step and ds is the space step.

5.2 Vorticity

5.2.1 Boundary conditions for the Vorticity

The boundary conditions for the vorticity stream line approach is quite complicated. The boundary conditions were formulated using the lecture notes (scan set 20) and equations 8-111 to 8-117 in the book. For our problem the boundary conditions are:-

At the bottom wall (j=1):

$$\Omega_{wall} = \{\Psi_{i,1} - \Psi_{i,2}\}\frac{2}{ds^2} + U_{wall}\frac{2}{ds} \quad \textbf{(6)}$$

At the top wall (j=ny):

$$\Omega_{wall} = \{\Psi_{i,ny} - \Psi_{i,ny-1}\}\frac{2}{ds^2} - U_{wall}\frac{2}{ds} \quad \textbf{(7)}$$

At the left wall (i=1)

$$\Omega_{wall} = \{\Psi_{1,j} - \Psi_{2,j}\}\frac{2}{ds^2} + U_{wall}\frac{2}{ds} \quad \textbf{(8)}$$

At the bottom wall (i=nx)

$$\Omega_{wall} = \{\Psi_{nx,i} - \Psi_{nx-1,j}\}\frac{2}{ds^2} - U_{wall}\frac{2}{ds} \quad \textbf{(9)}$$

5.3 Stream Function equation

Below is the poisson equation that is used for stream function and is referred as elliptic PDE. Once the stream function is calculated the velocity values (u,v) can be determined using equation 3.

$$\frac{\partial}{\partial x}\left(\frac{\partial \Psi}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{\partial \Psi}{\partial y}\right) = -\Omega \quad \textbf{(10)}$$

Discretized version of equation 10 is shown below dx=dy=ds

$$-\Omega = \frac{\Psi_{i+1,j,n} + \Psi_{i-1,j,n} + \Psi_{i,j+1,n} + \Psi_{i,j-1,n} - 4\Psi_{i,j,n}}{ds^2} \quad \textbf{(11)}$$

For solving the elliptical equation, we obtain,

$$\Psi_{i,j,n+1} = .25(\Psi_{i+1,j,n} + \Psi_{i-1,j,n} + \Psi_{i,j+1,n} + \Psi_{i,j-1,n} + ds*ds*\Omega_{i,j,n}) \quad \textbf{(12)}$$

5.4 Successive Over relaxation (SOR)

The method of successive over-relaxation (SOR) is a variant of the Gauss–Seidel method for solving a linear system of equations, resulting in faster convergence. A similar method can be used for any slowly converging iterative process.

$$\Psi_{i,j,n+1} = \beta*0.25\ (\Psi_{i+1,j,n} + \Psi_{i-1,j,n+1} + \Psi_{i,j+1,n} + \Psi_{i,j-1,n+1} + ds*ds*\Omega_{i,j,n}) + (1-\beta)_{i,j,n} \quad \textbf{(13)}$$

If $\beta=1$ in equation 13, then the method becomes gauss seidel method.
If $\beta>1$ in equation 13, method for the iteration is referred to over relaxation method.
If $\beta<1$ in equation 13, method for the iteration is referred to under relaxation method.

5.5 Pressure Calculation

The pressure was calculated using the streamline function. The pressure calculation in the stream vorticity approach uses the stream function to calculate the value of pressure at all the grid points. The equation used for the calculation of the pressure is shown below.

$$\nabla^2 P = 2*RHS$$

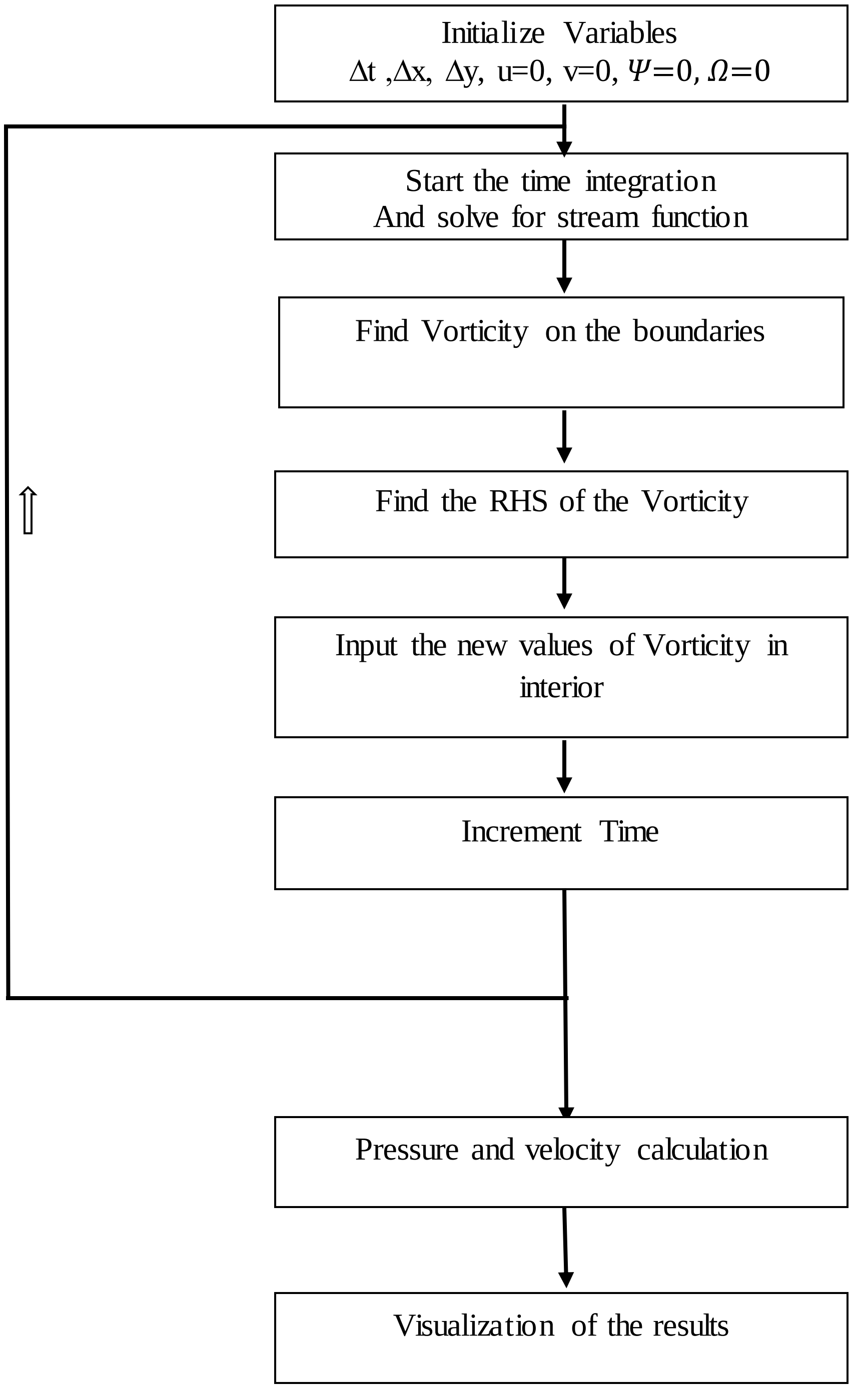
where, rhs is the right hand side of the pressure equation

$$RHS = \{\frac{\partial}{\partial x}\left(\frac{\partial \Psi}{\partial x}\right)\frac{\partial}{\partial y}\left(\frac{\partial \Psi}{\partial y}\right) - (\frac{\partial}{\partial y}\left(\frac{\partial \Psi}{\partial x}\right))^2\}$$

By plugging in the equation for rhs in the divergence of the pressure, the magnitude of the pressure is obtained.

6. DEVELOPMENT OF CODING ALGORITHM

The coding algorithm is illustrated with a block diagram. The code can be seen directly following the block diagram. The block diagram illustrates a short synopsis of how the code is employed to solve the vorticity equation.



7. RESULTS: NUMERICAL SIMULATION

Task I

Case I- Re1000 (taskI)

Shown below is the vorticity (a) and stream function (b) and stream function (with smaller contours levels) (c) in figure 2

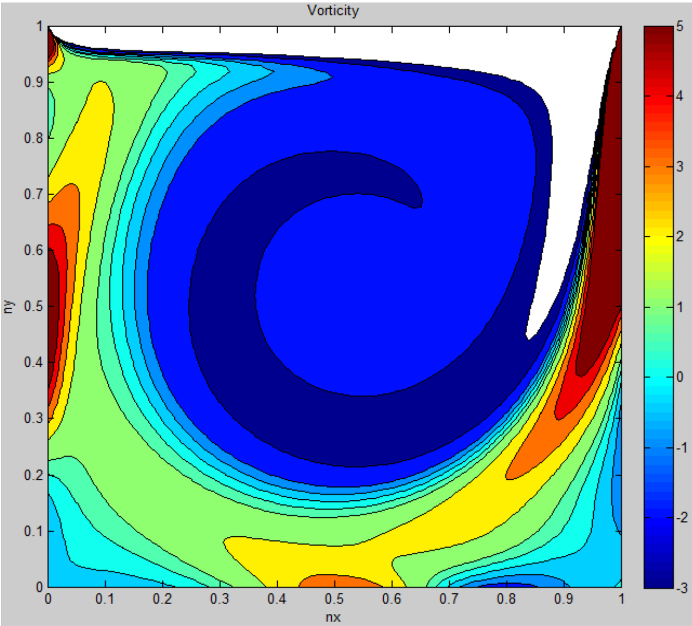
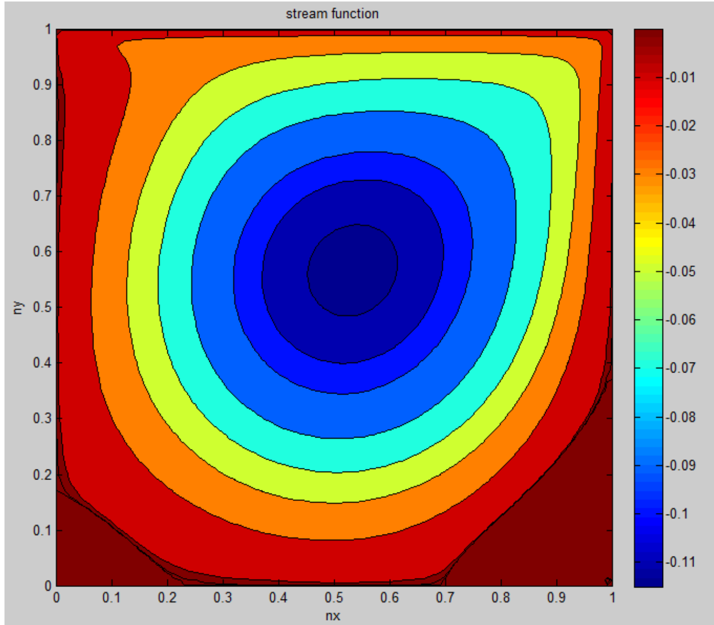


Figure 2. (a). Vorticity when Re =1000



(b). Steam Function when Re =1000

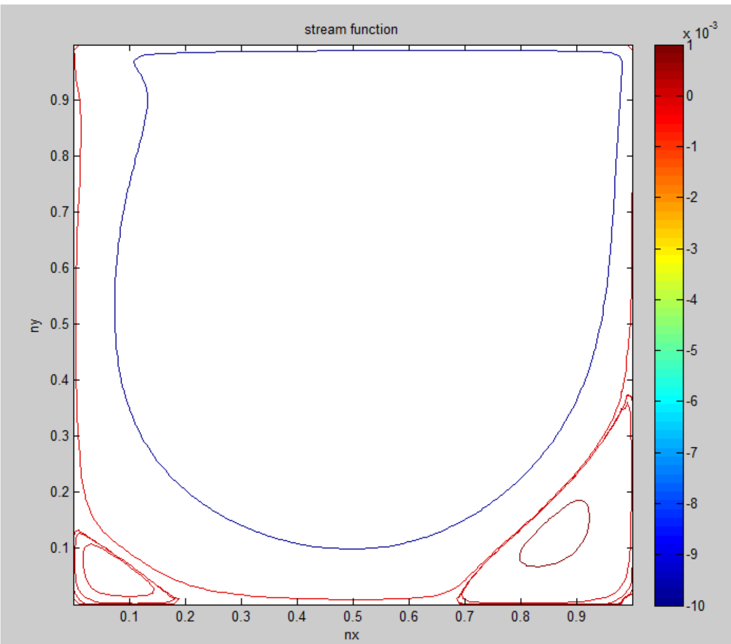


Figure 2 (c)

Shown below is the U velocity (a) and V velocity (b) plot in figure 3

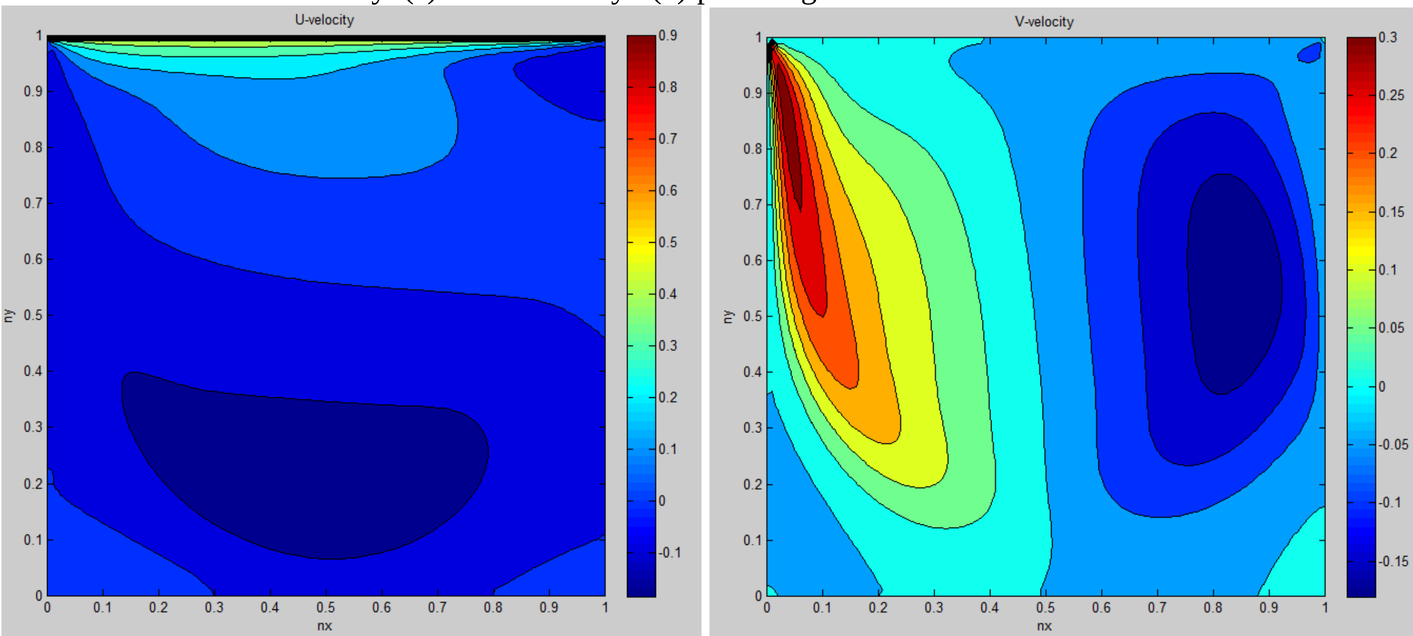


Figure 3. (a). U-velocity when Re =1000 (b). V-velocity when Re =1000

Case II- Re 100 (**task I**)

Shown below is the U velocity (a) and V velocity (b) plot in figure 4

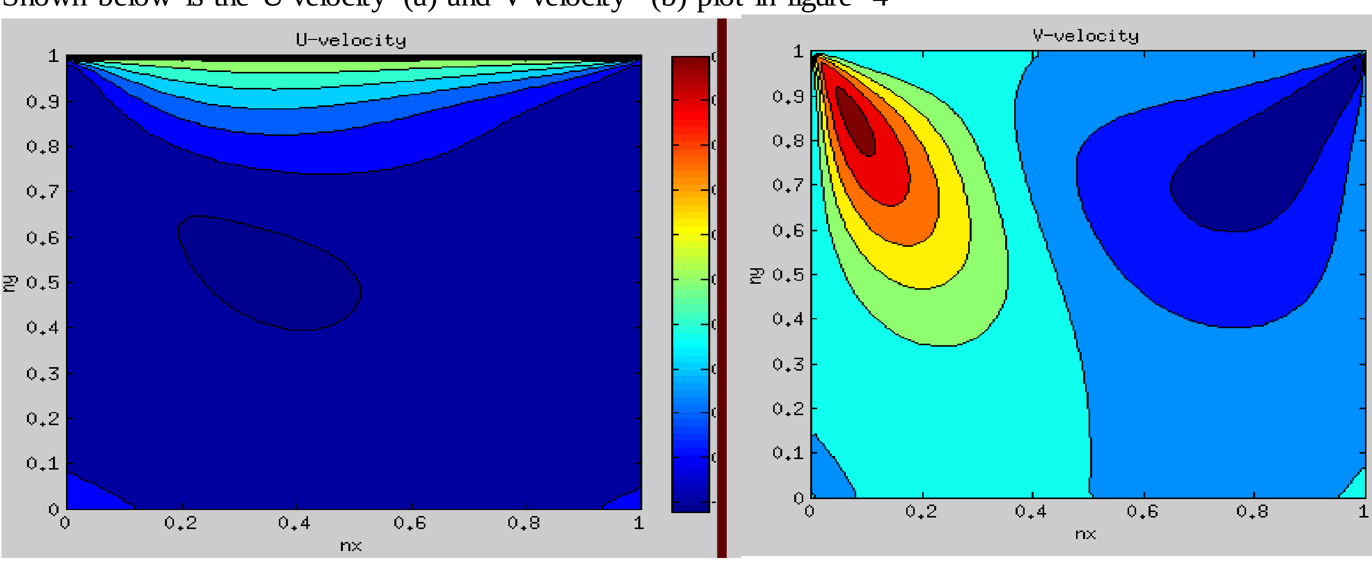


Figure 4. (a). U-velocity when Re =100 (b). V-velocity when Re =100

Shown below is the vorticity (a) and stream function (b) plot in figure 5

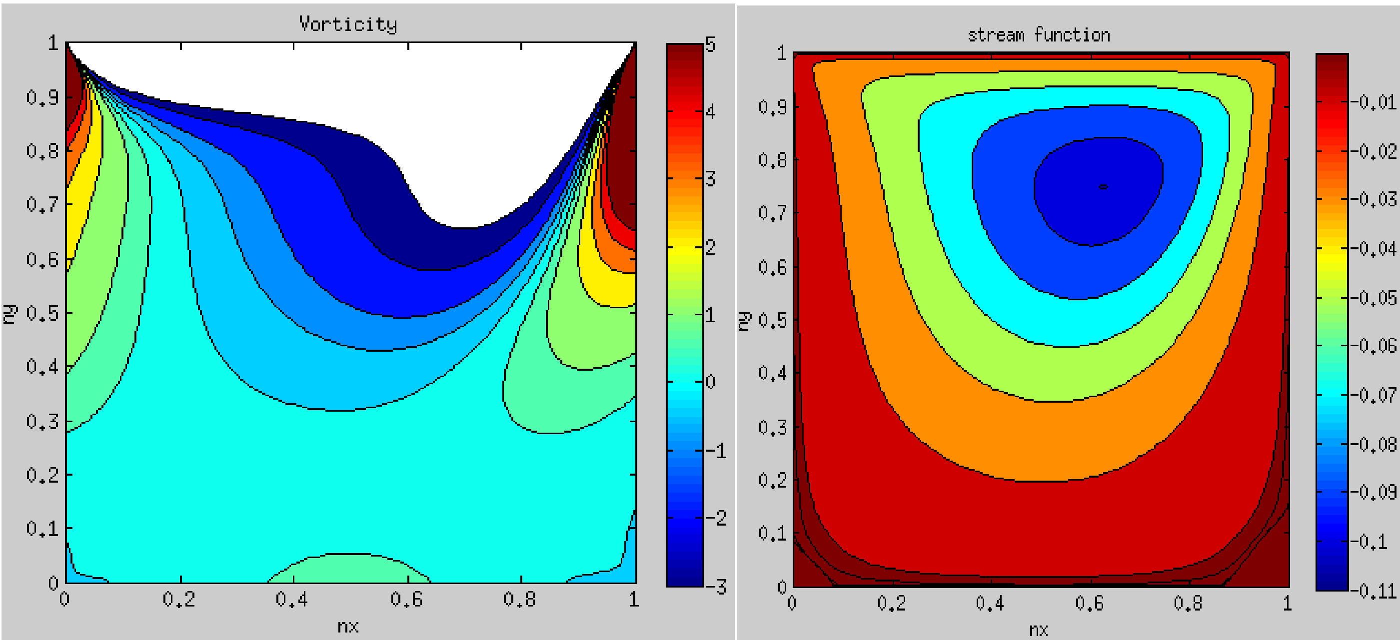


Figure 5. (a). Vorticity when Re =100 (b). Steam Function when Re =100

Task II

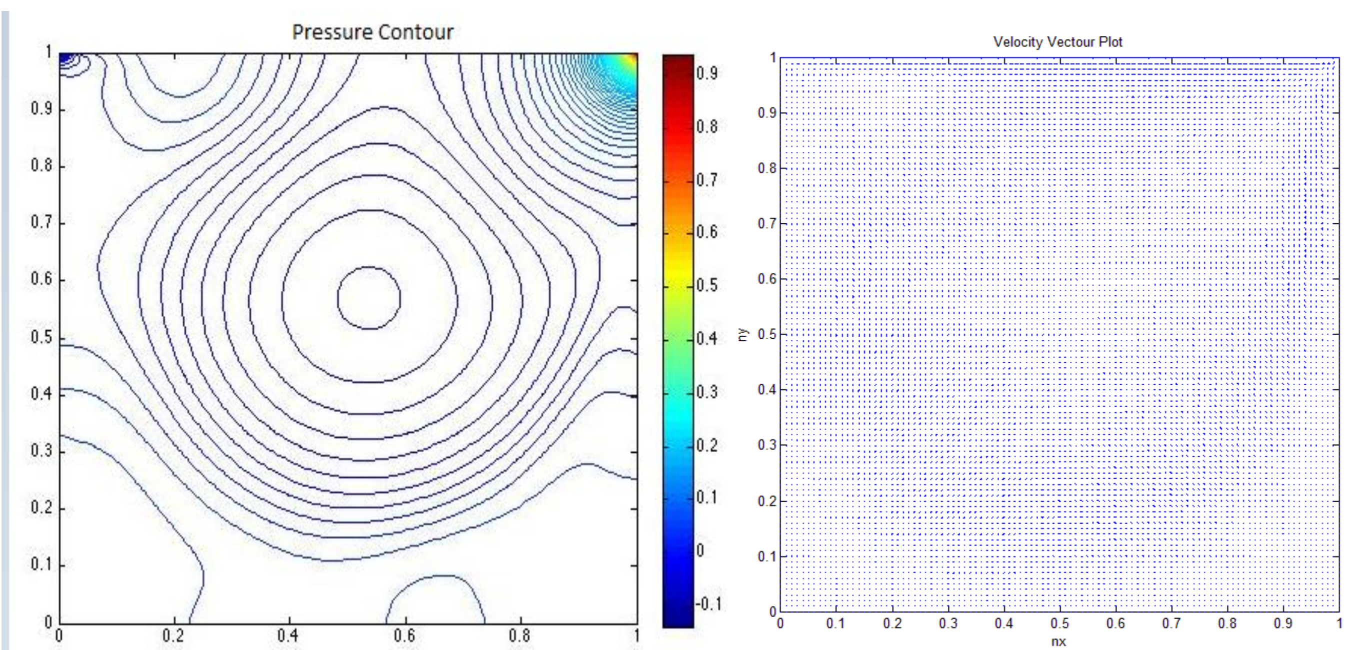


Figure 6. (a). Pressure contour when Re =1000 (b). Velocity Vector when Re =1000

Task III

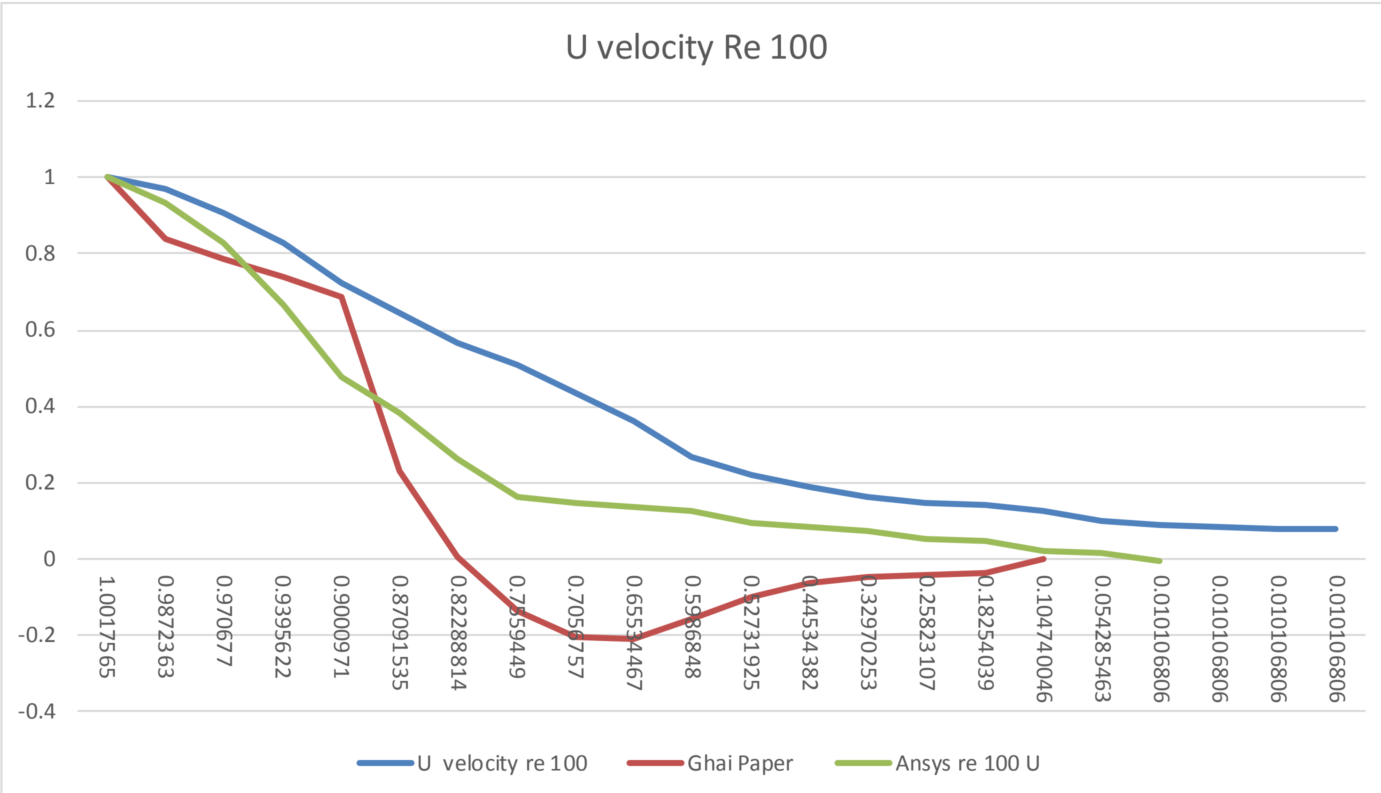


Figure 7. U- Velocity compassion/validation plots for Re=100.

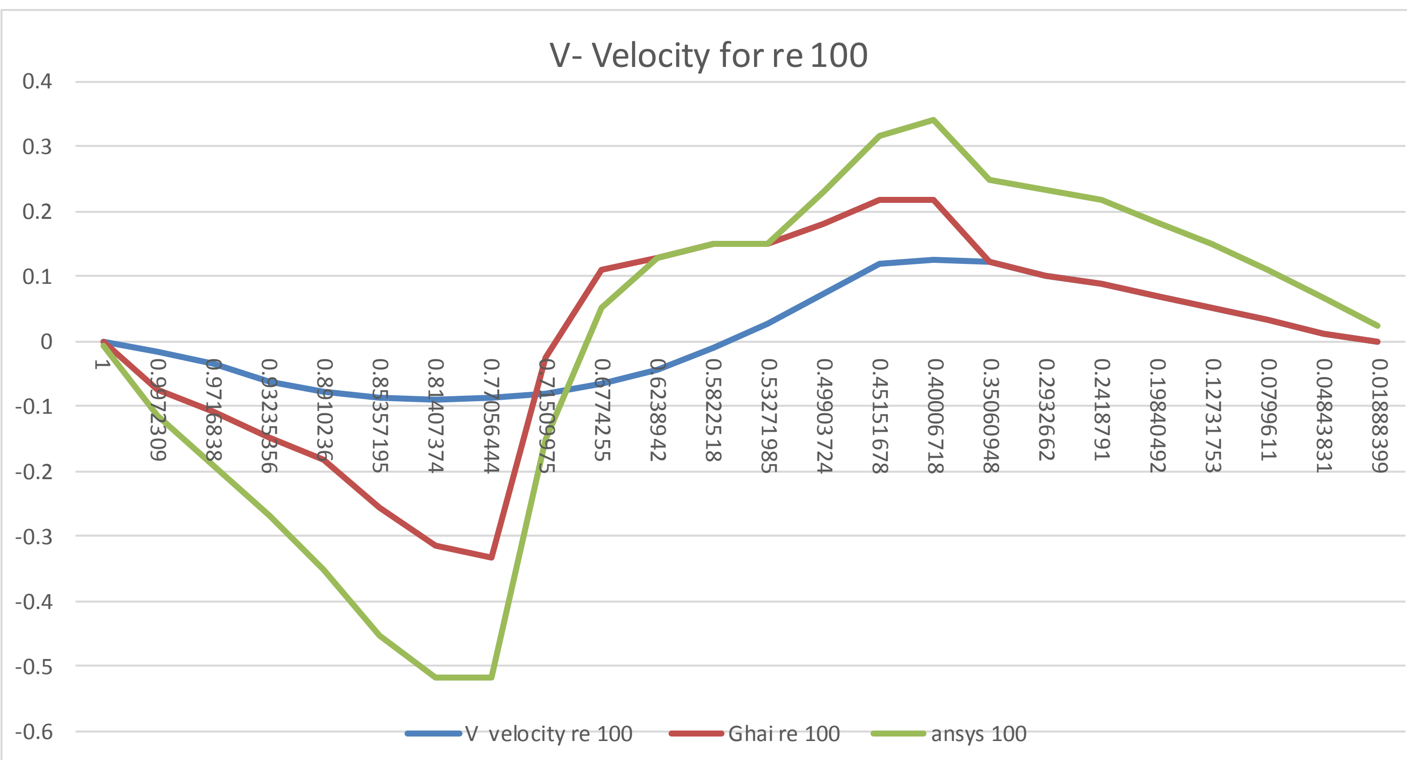


Figure 8. V- Velocity compassion/validation plots for Re=100.

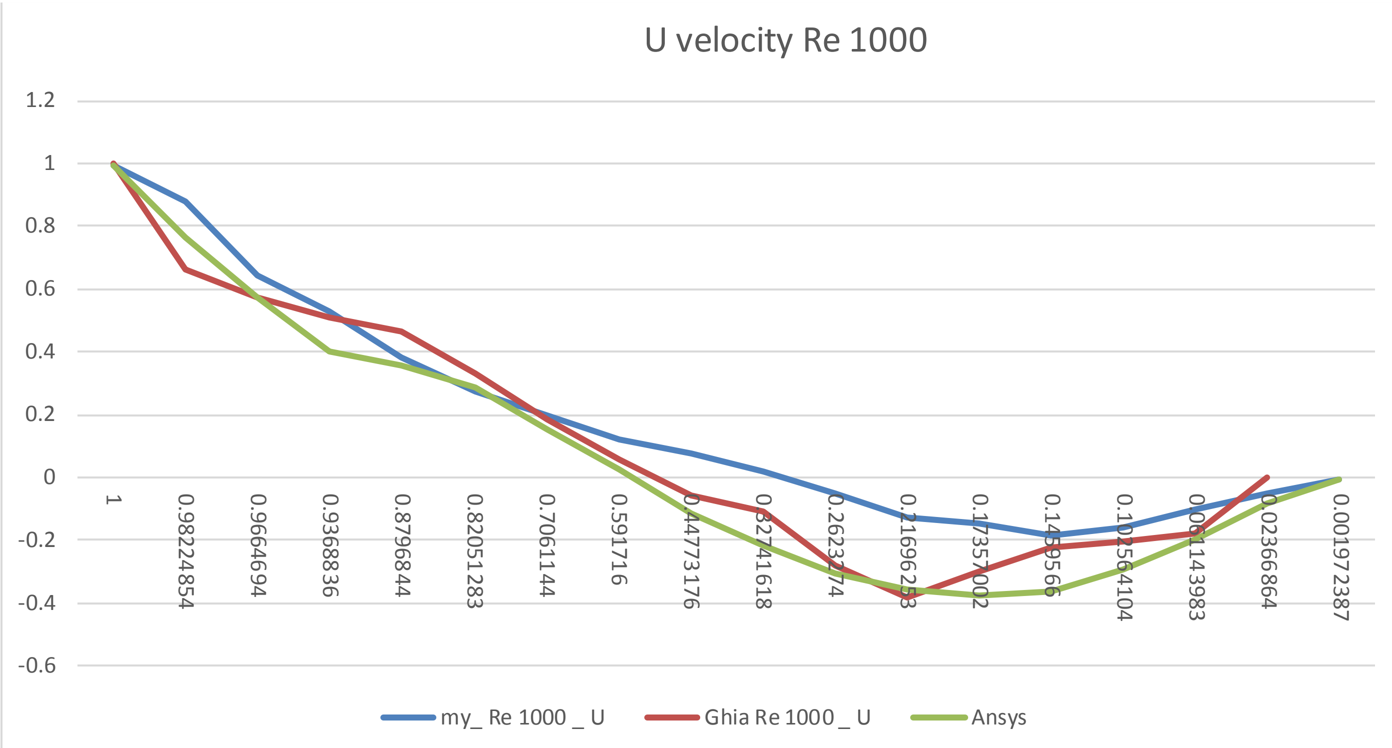


Figure 9. U- Velocity compassion/validation plots for Re=1000.

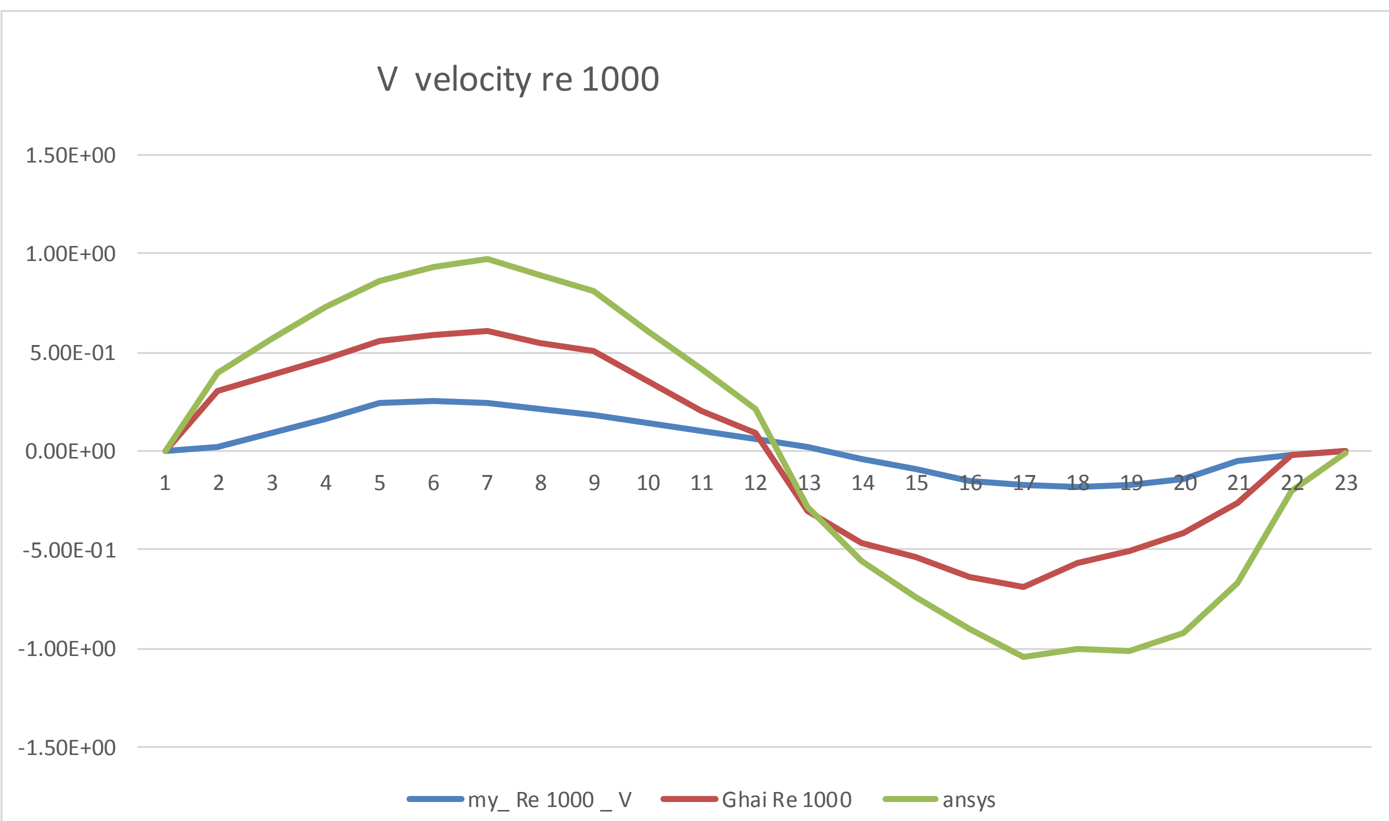


Figure 10. V- Velocity compassion/validation plots for Re=1000.

Note: The x axis in all the plots is the distance (x or y distance) and Y axis is the velocity (u or v)

Task IV

Shown below is the vorticity (a) and stream function (b) plot in figure11

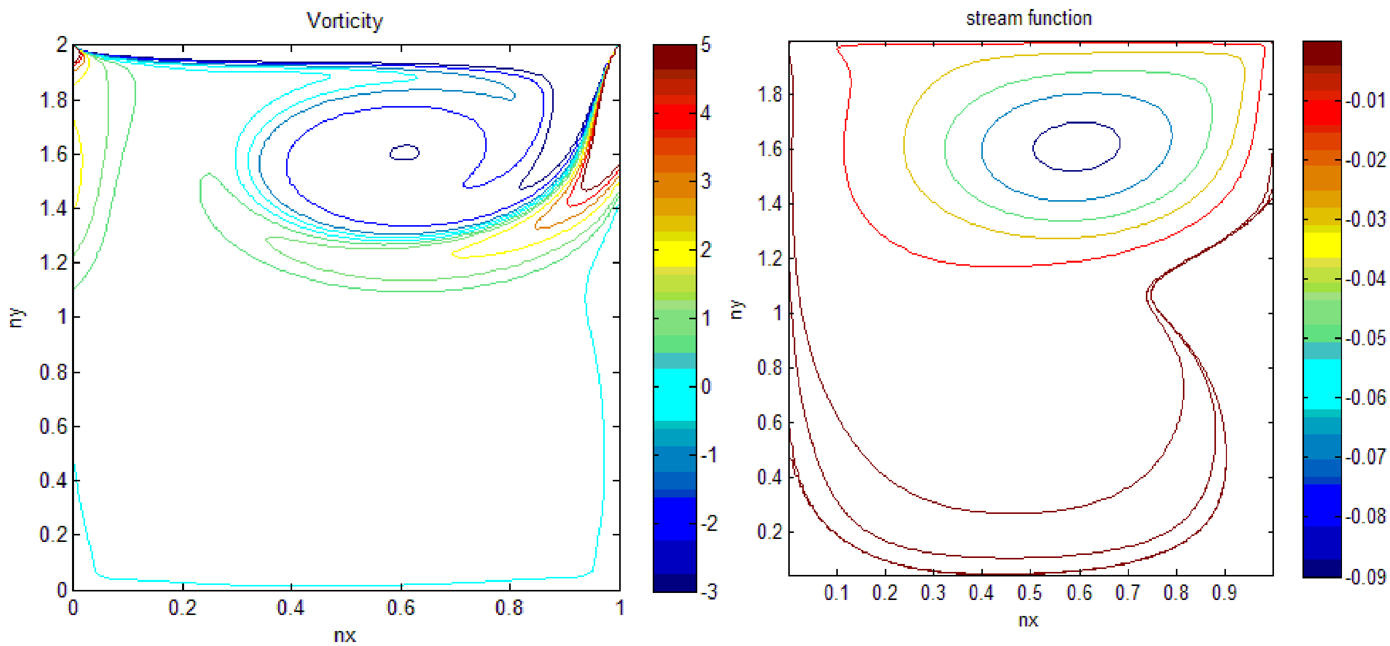


Figure 11. (a). Vorticity when AR=2 (b). Steam Function when AR=2

Shown below is the U velocity (a) and V velocity (b) plot in figure 12

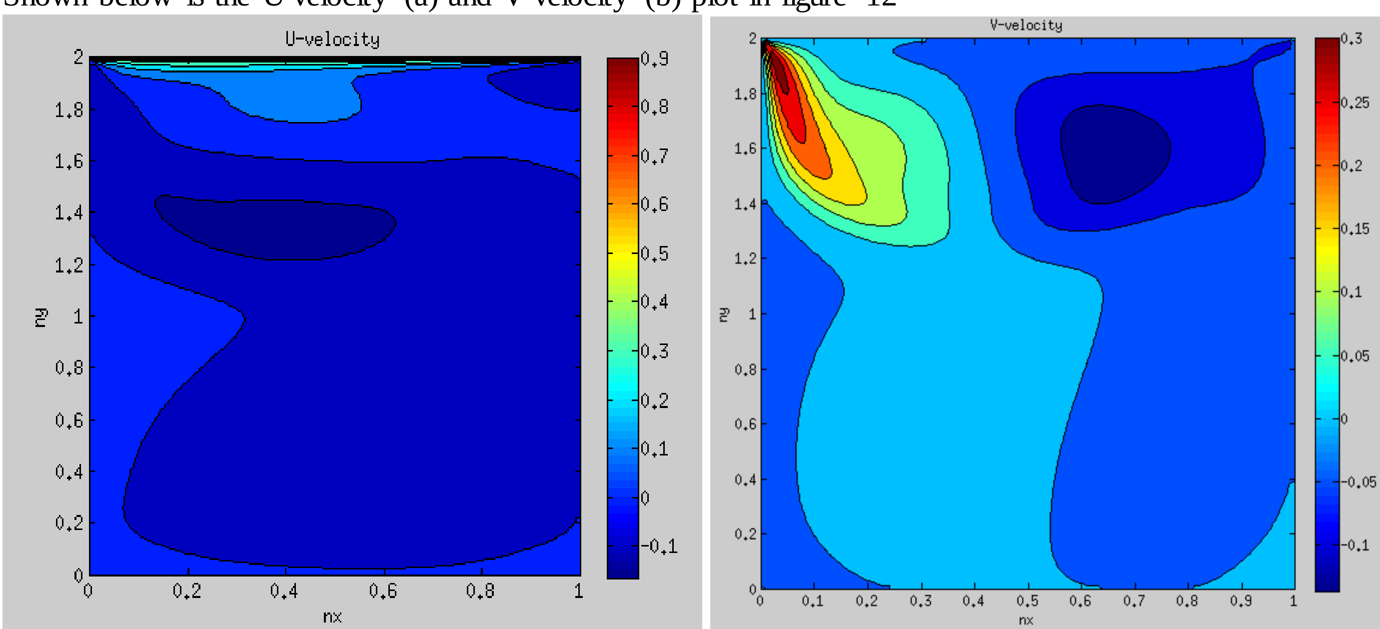
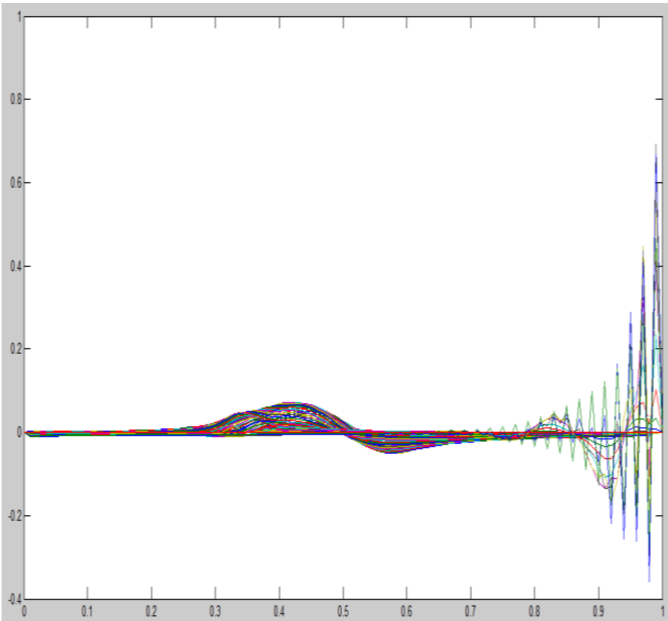
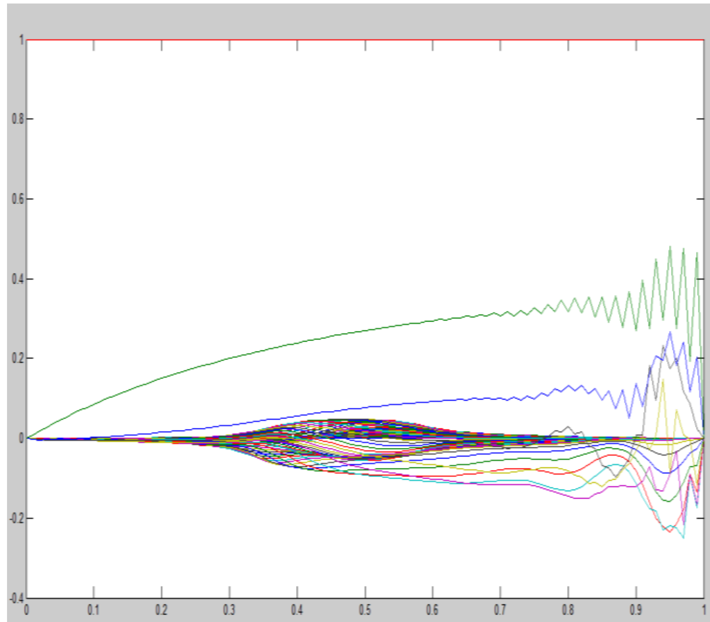


Figure 12 (a). U-velocity when AR=2 (b). V-velocity when AR = 2

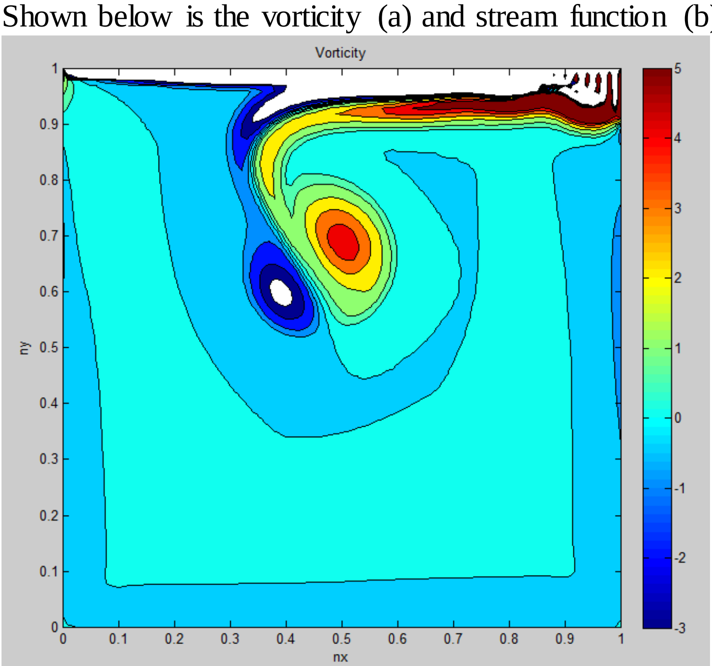
Task 5

Reynolds Number = 10000 (task 5)





Shown below is the vorticity (a) and stream function (b) plot in figure 13



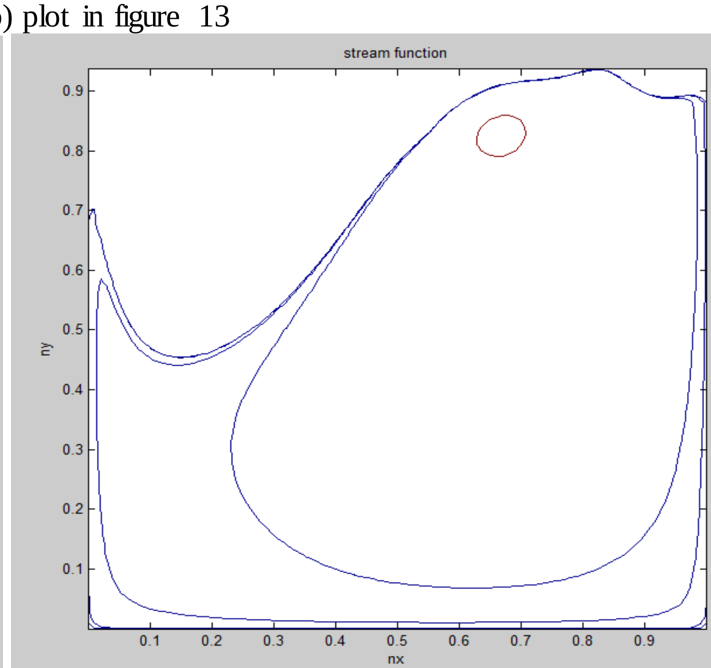
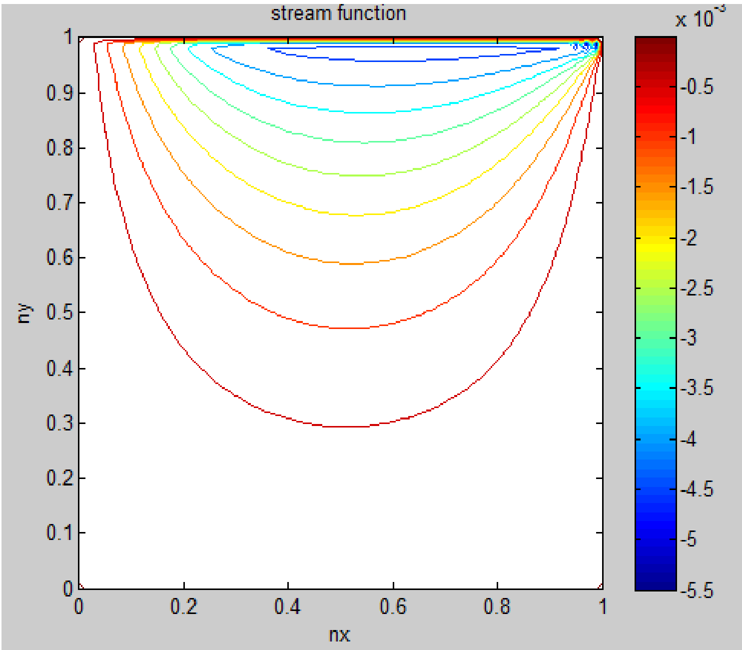
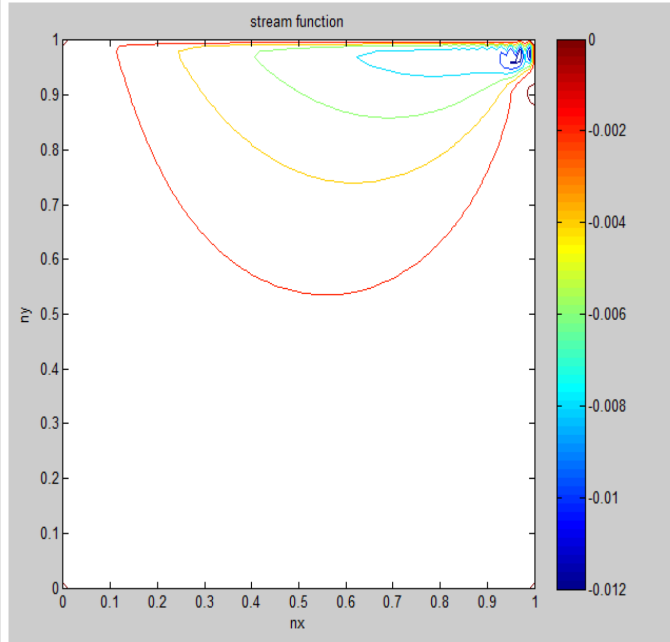


Figure 13. (a). Vorticity when Re =10000 (b). Steam Function when Re =10000

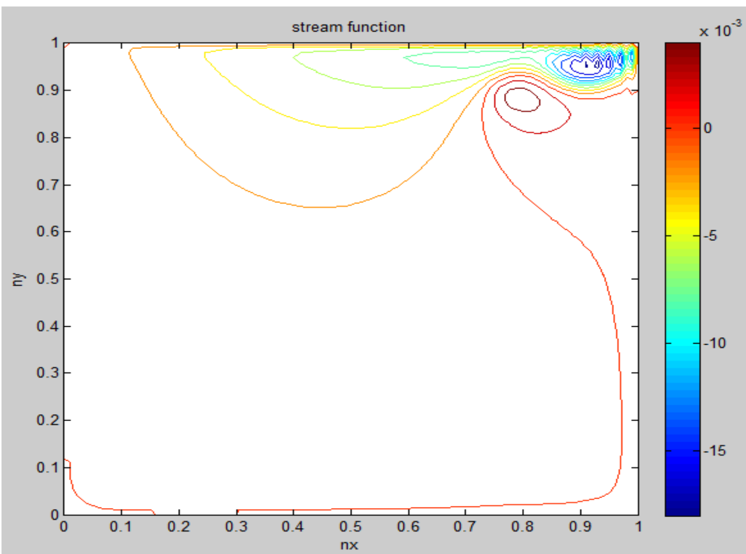
Shown below is the U velocity (a) and V velocity (b) plot in figure 14



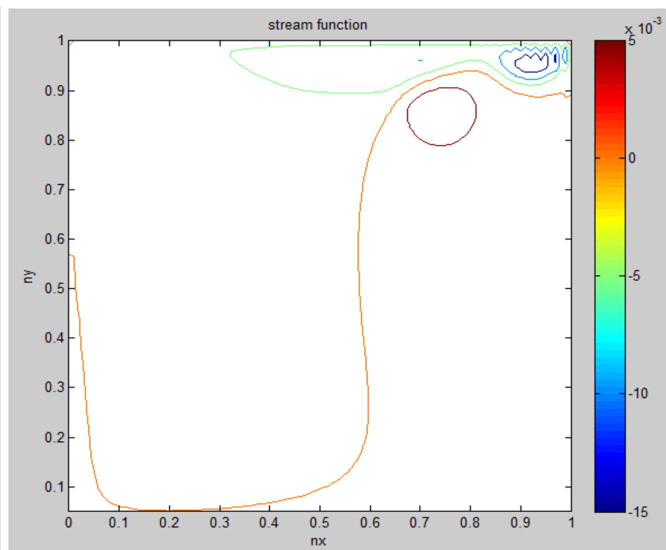
(c). Steam Function when Re =10000 (t=0)



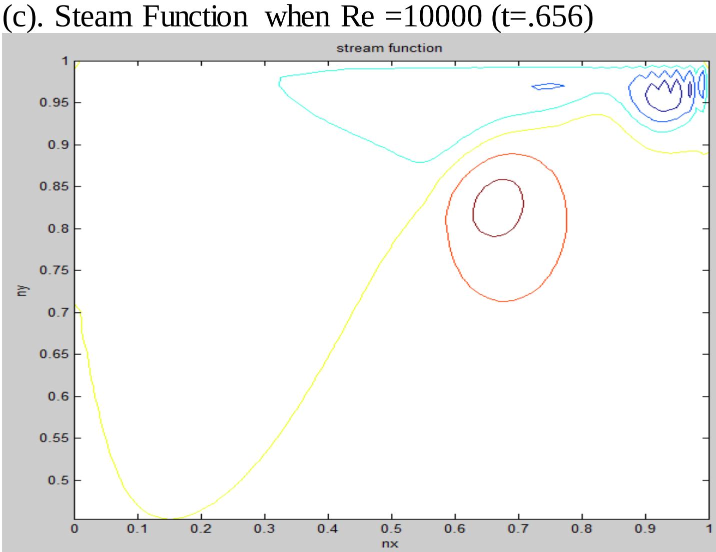
(c). Steam Function when Re =10000 (t=0.328)

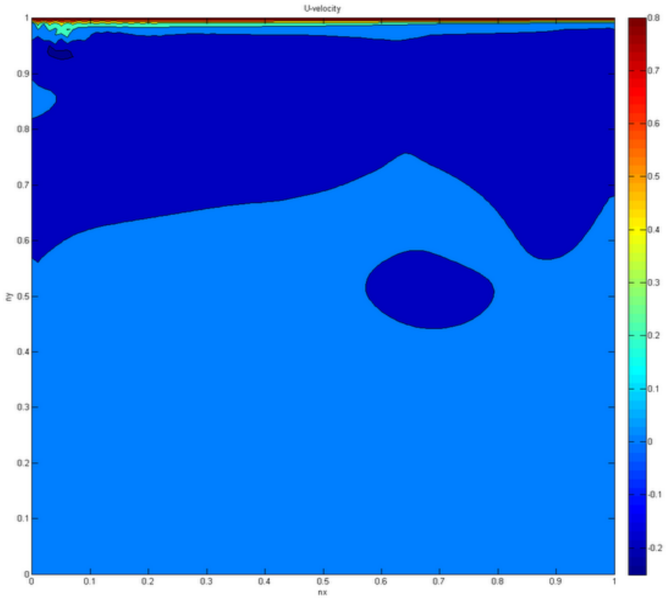


(c). Steam Function when Re =10000 (t=.656)



(c). Steam Function when Re =10000 (t=0.984)





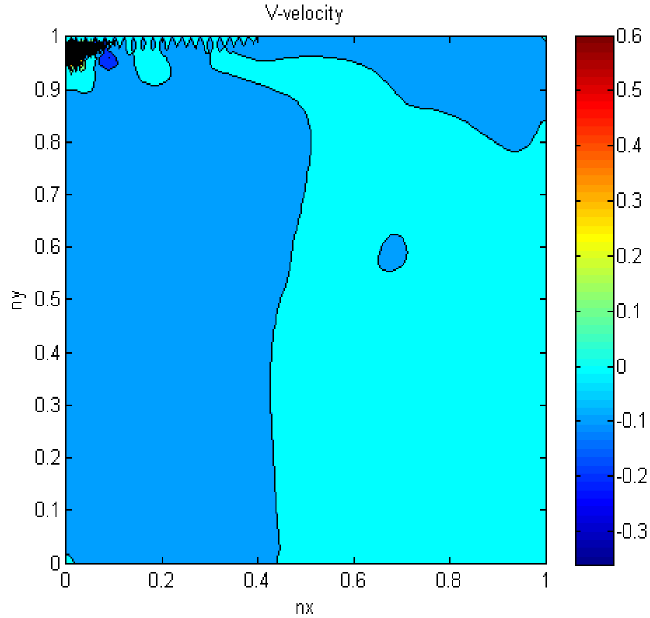


Figure 14. (a). U-velocity when Re =1000 (b). V-velocity when Re =1000

7.1 RESULTS ANSYS : FLUENT RESULTS

1) Case I - Reynolds Number 1000

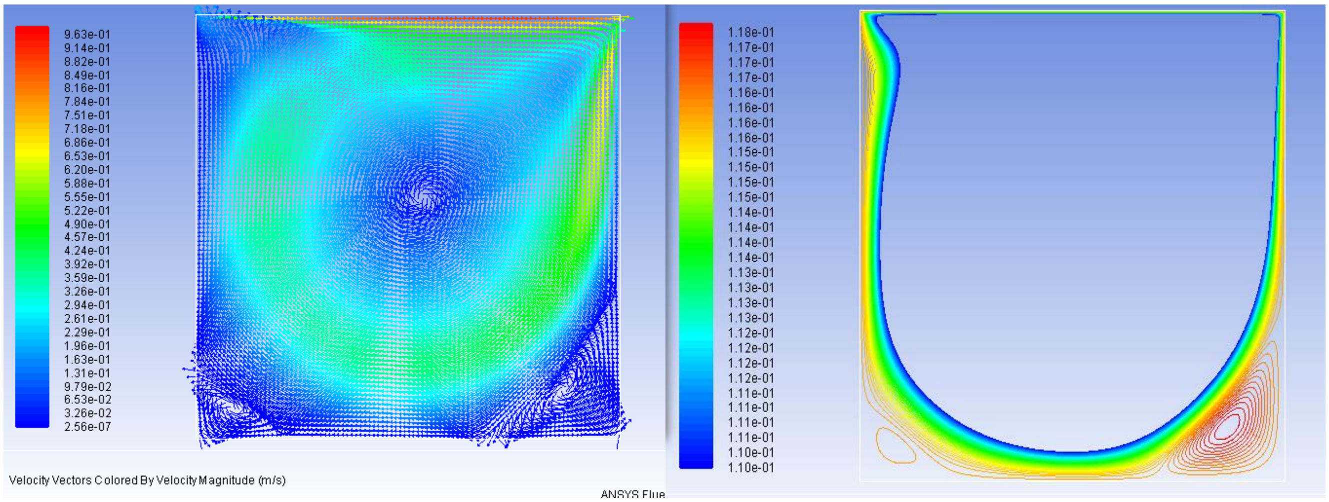


Figure 16. (a). Velocity vector for Re 1000 (b). Steam Function when Re =1000 (with vortices) [4]

To get the corner vortices , the contour levels are adapted from [4].

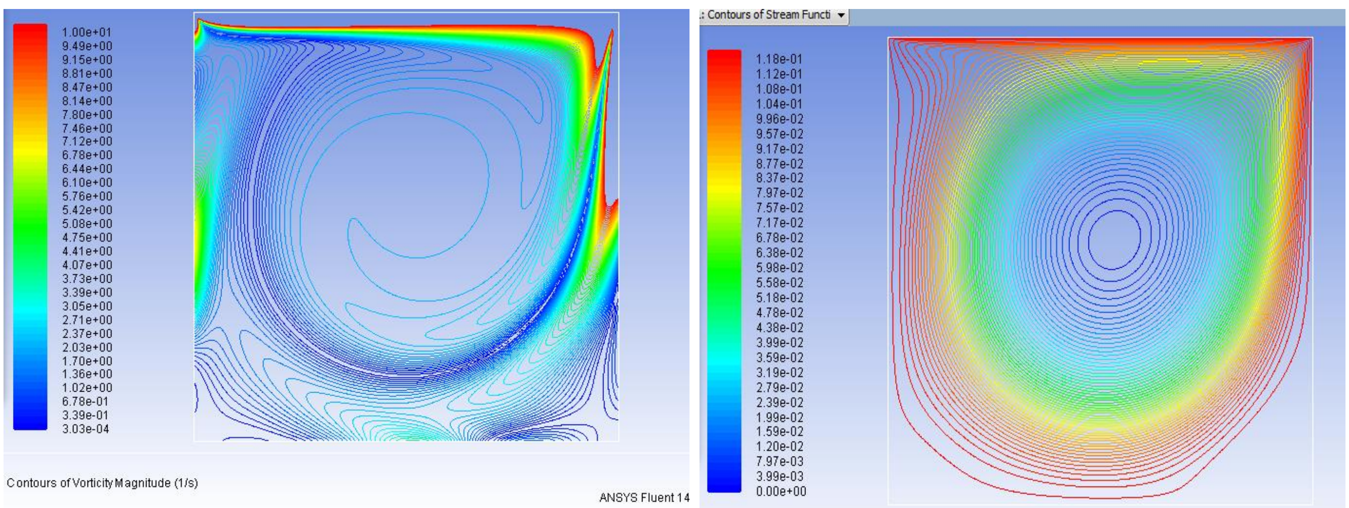
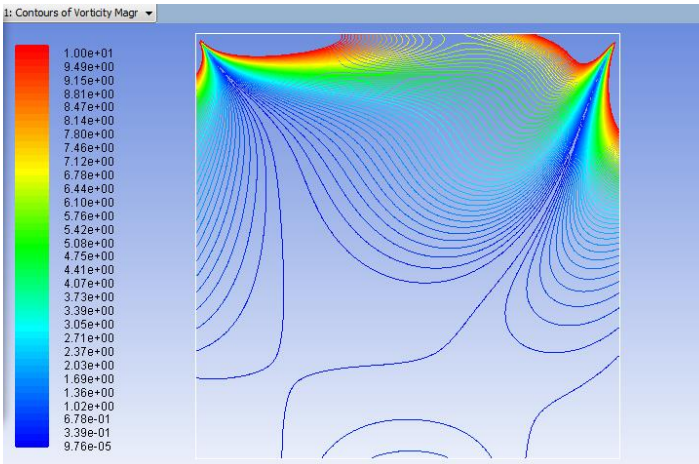


Figure17 (a). Vorticity when Re =1000 (b). Steam Function when Re =1000

Case I - Reynolds Number 100



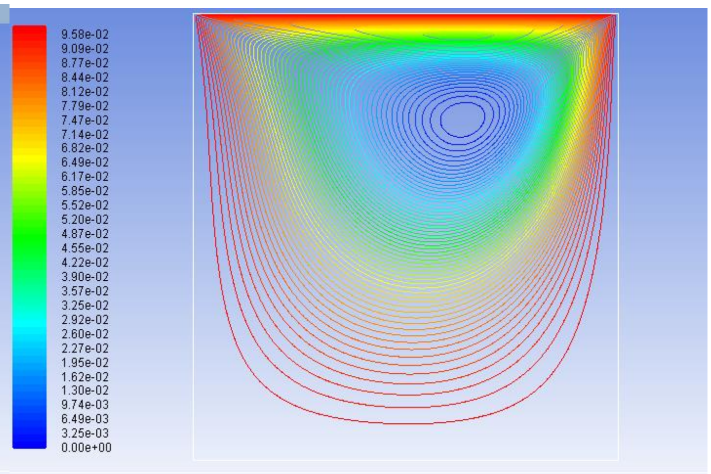
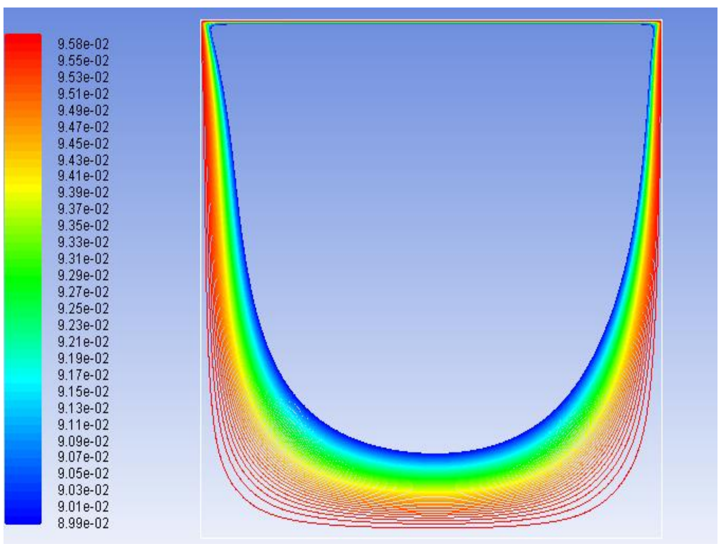


Figure 18. (a). Vorticity when Re =100 (b). Steam Function when Re =100



(c) Steam Function when Re =1000 (small contours)

9. CONCLUSIONS

This project was the summary of what was learnt in this course during this semester. We started with the 1-D problems and moved toward the more complicated problems. The last homework was particularly helpful in finding the solutions of this project. This project closely resemble actual engineering problems and thus an important aspect of MAE 561 Computational fluid dynamics. The vorticity equation and a single moving wall the fluid was driven in a circular path. The contours for the vorticity and streamline function was presented for both the aspect ratio 1(task I) as well as 2(task IV). The velocity contour and pressure contour was also presented for re 1000 (task II). This is an intuitive solution; however, a precise solution would be extremely difficult, if not impossible, because it depends on the grid size, space step and time steps and many other factors.

10. REFERENCES

[1]Arakawa, A., 1966: Computational design for long-term numerical integration of teh equations of fluid motion

[2]Two-dimensional incopressible flow, Part I, Journal of Computational Physics, 1, 119-143 Bruneau, C.-H., and M. Saad, 2006: The 2-D cavity driven flow revisited, Computers and Fluids, 35,
326-348

[3]Ghia, U., K. N. Ghia, and C. T. Shin, 1982: High-Re solutions of incompressible flow using the Navier-Stokes equations and a multigrid method, Journal of Computational Physics, 48, 387-411

[4] Flow in a Lid-Driven Cavity, http://cfd.iut.ac.ir/files/cavity.pdf.

11. APPENDIX

```
%Author AKSHAY BATRA
 %MAE 561 Computational Fluid Dynamics
 %FINAL PROJECT - LID DRIVEN CAVITY FLOW IN RECATANGULAR CAVITY
 %Due on Dec 12 2014.
 %-----------------------------------------------------------------------
 clear all
 close all
 clf;
 nx=101;ny=101; nt=100000; re=1000; dt=0.001;% Settting the initial parameters
 no_it=100000;% number of iterations
 Beta=1.5;% relaxation factors
 err=0.001;% parameter for SOR iteration
 ds=.01;%dx=dy=ds
 x=0:ds:1; y=0:ds:1;%dimensions of the cavity
 t=0.0;
 %-----------------------------------------------------------------------
 phi=zeros(nx,ny); omega=zeros(nx,ny); % initializing the variables
 u = zeros(nx,ny); v = zeros(nx,ny);
 x2d=zeros(nx,ny); y2d=zeros(nx,ny);
 b=zeros(nx,ny);p=zeros(nx,ny);pn=zeros(nx,ny);
 w=zeros(nx,ny); %p-q/(nx-1),

 %-----------------------------------------------------------------------
 %Stream Function calculation
for t_step=1:nt % time steps starts
   for iter=1:no_it % streamfunction calculation
   w=phi; % by SOR iteration
       for i=2:nx-1;
           for j=2:ny-1
 phi(i,j)=0.25*Beta*(phi(i+1,j)+phi(i-1,j)+phi(i,j+1)+phi(i,j-1)+ds*ds*omega(i,j))+(1.0-
Beta)*phi(i,j);
           end
       end
 Err=0.0;
 for i=1:nx
     for j=1:ny
         Err=Err+abs(w(i,j)-phi(i,j));
     end
 end
     if Err <= err,
     break;
     end % stop if iteration has converged
   end
%-----------------------------------------------------------------------
%boundary conditions for the Vorticity
for i=2:nx-1
    for j=2:ny-1
 omega(i,1)=-2.0*phi(i,2)/(ds*ds);                              % bottom wall
 omega(i,ny)=-2.0*phi(i,ny-1)/(ds*ds)-2.0/ds;                   % top wall
 omega(1,j)=-2.0*phi(2,j)/(ds*ds);                              % right wall
 omega(nx,j)=-2.0*phi(nx-1,j)/(ds*ds);                          % left wall
    end
```

```
end

%----------------------------------------------------------------------------
% RHS Calculation
 for i=2:nx-1;
     for j=2:ny-1 % compute
 w(i,j)=-0.25*((phi(i,j+1)-phi(i,j-1))*(omega(i+1,j)-omega(i-1,j))...
 -(phi(i+1,j)-phi(i-1,j))*(omega(i,j+1)-omega(i,j-1)))/(ds*ds)...
+(1/re)*(omega(i+1,j)+omega(i-1,j)+omega(i,j+1)+omega(i,j-1)-4.0*omega(i,j))/(ds*ds);
     end
 end
 %----------------------------------------------------------------------------
% Update the vorticity
 omega(2:nx-1,2:ny-1)=omega(2:nx-1,2:ny-1)+dt*w(2:nx-1,2:ny-1);

 t=t+dt; % increment the time
 for i=1:nx
    for j=1:ny
    x2d(i,j)=x(i);
    y2d(i,j)=y(j);
    end
 end
 %---------------------------------------------------------------------------
 %calculation of U and V
    for i = 2:nx-1
        for j = 2:ny-1
u(i,j)=(phi(i,j+1)-phi(i,j))/(2*ds);
v(i,j)=(phi(i+1,j)-phi(i,j))/(2*ds);
u(:,ny) = 1;
v(nx,:) =.02;
        end
    end
end
%----------------------------------------------------------------------------
%calculation of pressure
rhs=zeros(nx,ny);
for i=2:nx-1
    for j=2:ny-1
        rhs(i,j)=(((phi(i-1,j)-2*phi(i,j)+phi(i+1,j))/(ds*ds))...
            *((phi(i,j-1)-2*phi(i,j)+phi(i,j+1))/(ds*ds)))...
            - (phi(i+1,j+1)-phi(i+1,j-1)-phi(i-1,j+1)+phi(i-1,j-1))/(4*(ds*ds));

   p(i,j)=(.25*(pn(i+1,j)+pn(i-1,j) + pn(i,j+1)+pn(i,j-1))- 0.5*((rhs(i,j)*ds^2*ds^2)));

    end
    pn=p;

end

%----------------------------------------------------------------------------
% Visualization of the results
 figure(1)
contourf(x2d,y2d,omega,[-3:1:-1 -0.5 0.0 0.5 1:1:5 ]),xlabel('nx'),...
    ylabel('ny'),title('Vorticity');axis('square','tight');colorbar
title('Vorticity') % plot vorticity

figure(2)
contour(x2d,y2d,phi,[10^-10  10^-7 10^-5 10^-4 0.0100...
0.0300 0.0500 0.0700 0.0900 0.100 0.1100 0.1150 0.1175]),xlabel('nx'),
```

```
ylabel('ny'),title('stream function');axis('square','tight');colorbar %streamfunction

figure(3)
contourf(x2d,y2d,u),xlabel('nx'),ylabel('ny'),...
    title('U-velocity');axis('square','tight');colorbar

figure(4)
contourf(x2d,y2d,v),xlabel('nx'),ylabel('ny'),...
    title('V-velocity');axis('square','tight');colorbar

figure(5)
contourf(x2d,y2d,p,([-2.0:.01:2])),xlabel('nx'),ylabel('ny'),...
    title('pressure');
figure(6)
quiver (x2d,y2d,u,v)...
 ,xlabel('nx'),ylabel('ny'),title('Velocity Vectour Plot'); axis([0 1 0
1]),axis('square')

%--------------------------------------------------------------------------
%I have been able to get the vorticies in the stream function contour at the corner but
%they are really small(for re 100). I have run the solution to 100,000 iteration at a
%time step of .001. It took 9-10 hours for the solution to compute.Then
%similarly for the Re 1000 took even longer  but i was able to get the vorticies.
```