NATIONAL INSTITUE OF TECHNOLOGY KARANATAKA, SURATHKAL

ASSIGNMENT 13

APPLIED COMPUTATIONAL METHODS IN MECHANICAL SCIENCES

RAJAT A CHANDAVAR – 16ME156

13-Nov-19

ASSIGNMENT ON LID DRIVEN CAVITY

Answer

- All values computed are non-dimensional values.
- Validation of results were done with Ghia et. al. [1]
- All property calculation were done at node points.

Code(C++)

```
#include<iostream>
#include<cmath>
#include<fstream>
using namespace std;
main()
{
    fstream pr,x_vel,y_vel,er,data;
    pr.open("Pressure.txt",ios::out);
    x_vel.open("x-velocity.txt",ios::out);
    y_vel.open("y-velocity.txt",ios::out);
    er.open("Error.txt",ios::out);
    data.open("data.dat",ios::out);
    int i,j,no_x=128,no_y=128,iter=0;
    double pn[no_y+2][no_x+2]={ },p[no_y+2][no_x+2]={ },u[no_y+2][no_x+1]={ },v[no_y+1][no_x+2]={ };
    double F[no_y+2][no_x+1]={ },G[no_y+1][no_x+2]={ },p_error,max_p_error,v_error,u_error,max_v_error,max_u_error,velocity_error;
    double dx=1.0/no_x,dy=1.0/no_y,r=dx/dy,dt=0.001,Re=100,time;
    double ue,uw,un,us,ve,vw,vn,vs,ae,ap,aw,an,as,q,omega_sor=1.9,temp;
    do
    {
        ++iter;
        time=dt*iter;
        //y-velocity Boundary conditions
        for(j=1;j<no_x+1;++j)
        {
            v[0][j]=0.0;//Bottom
            v[no_y][j]=0.0;//Top
        }
        for(i=0;i<no_y+1;++i)
        {
            v[i][0]=-v[i][1];//Left
            v[i][no_x+1]=-v[i][no_x];//Right
        }
        //x-velocity Boundary conditions
        for(i=1;i<no_y+1;++i)
        {
            u[i][0]=0.0;//Left
            u[i][no_x]=0.0;//Right
        }
        for(j=0;j<no_x+1;++j)
        {
            u[0][j]=-u[1][j];//Bottom
            u[no_y+1][j]=2-u[no_y][j];//Top
        }
        //Calculation of x-momentum coefficient F
        for(i=1;i<no_y+1;++i)
        {
            for(j=1;j<no_x;++j)
            {
                ue=(u[i][j]+u[i][j+1])/2;
                uw=(u[i][j-1]+u[i][j])/2;
                un=(u[i][j]+u[i+1][j])/2;
                us=(u[i][j]+u[i-1][j])/2;
                vn=(v[i][j]+v[i][j+1])/2;
                vs=(v[i-1][j+1]+v[i-1][j])/2;
```

```
                    F[i][j]=-((ue*ue-uw*uw)/dx+(un*vn-us*vs)/dy-
(1.0/Re)*((u[i][j+1]-2*u[i][j]+u[i][j-1])/(dx*dx)+(u[i+1][j]-2*u[i][j]+u[i-
1][j])/(dy*dy)));
                }
            }
            //Calculation of y-momentum coefficient G
            for(i=1;i<no_y;++i)
            {
                for(j=1;j<no_x+1;++j)
                {
                    vn=(v[i+1][j]+v[i][j])/2;
                    vs=(v[i][j]+v[i-1][j])/2;
                    ve=(v[i][j]+v[i][j+1])/2;
                    vw=(v[i][j]+v[i][j-1])/2;
                    ue=(u[i+1][j]+u[i][j])/2;
                    uw=(u[i+1][j-1]+u[i][j-1])/2;
                    G[i][j]=-((vn*vn-vs*vs)/dy+(ue*ve-uw*vw)/dx-
(1.0/Re)*((v[i+1][j]-2*v[i][j]+v[i-1][j])/(dy*dy)+(v[i][j+1]-2*v[i][j]+v[i][j-
1])/(dx*dx)));
                }
            }
            do//SOR for Pressure
            {
                for(i=no_y;i>0;--i)//computing from top
                {
                    for(j=1;j<no_x+1;++j)
                    {
                        aw=1.0;
                        ae=1.0;
                        an=r*r;
                        as=r*r;
                        q=((F[i][j]-F[i][j-1])/dx+(G[i][j]-G[i-1][j])/dy)*dx*dx;
                        if(i==1)
                            as=0;
                        else if(i==no_y)
                            an=0;
                        if(j==1)
                            aw=0;
                        else if(j==no_x)
                            ae=0;
                        ap=-(aw+ae+as+an);
                        pn[i][j]=(1-omega_sor)*p[i][j]+omega_sor*(q-ae*pn[i][j+1]-
aw*pn[i][j-1]-an*pn[i+1][j]-as*pn[i-1][j])/ap;
                        p_error=abs(pn[i][j]-p[i][j]);
                        if(i==no_y && j==1)
                            max_p_error=p_error;
                        else if(p_error>max_p_error)
                            max_p_error=p_error;
                    }
                }
                //Update pressure
                for(i=0;i<no_y+2;++i)
                    for(j=0;j<no_x+2;++j)
                        p[i][j]=pn[i][j];
            }while(max_p_error>1e-6);
            for(i=1;i<no_y+1;++i)
            {
                for(j=1;j<no_x;++j)
                {
                    temp=u[i][j];
                    u[i][j]=u[i][j]+dt*F[i][j]-(dt/dx)*(p[i][j+1]-p[i][j]);
                    u_error=abs(u[i][j]-temp);
                    if(i==1 && j==1)
                        max_u_error=u_error;
                    else if(u_error>max_u_error)
                        max_u_error=u_error;
                }
            }
```

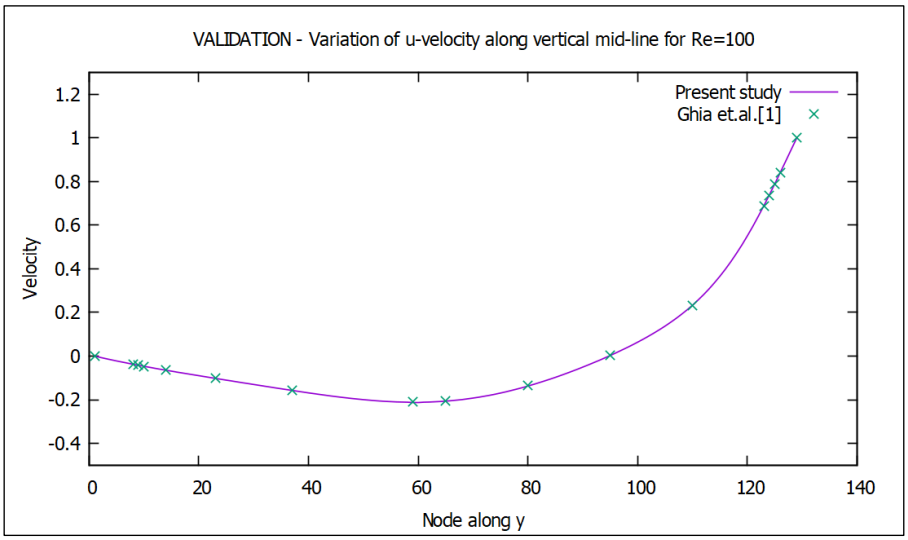
```
        for(i=1;i<no_y;++i)
        {
            for(j=1;j<no_x+1;++j)
            {
                temp=v[i][j];
                v[i][j]=v[i][j]+dt*G[i][j]-(dt/dy)*(p[i+1][j]-p[i][j]);
                v_error=abs(v[i][j]-temp);
                if(i==1 && j==1)
                    max_v_error=v_error;
                else if(v_error>max_v_error)
                    max_v_error=v_error;
            }
        }
        if(max_u_error>max_v_error)
            velocity_error=max_u_error;
        else
            velocity_error=max_v_error;
        cout<<"Iteration:"<<iter<<" "<<"Error:"<<velocity_error<<"\n";
        er<<time<<" "<<u[no_y/2][no_x/2]<<"\n";
    }while(velocity_error>1e-8);
    er.close();
    //Final solution Pressure boundary conditions
    for(i=1;i<no_y+1;++i)
    {
        pn[i][0]=pn[i][1];
        pn[i][no_x+1]=pn[i][no_x];
    }
    for(j=0;j<no_x+2;++j)
    {
        pn[0][j]=pn[1][j];
        pn[no_y+1][j]=pn[no_y][j];
    }
    //Final solution Velocity boundary conditions
    for(i=0;i<no_y+1;++i)
    {
        v[i][0]=-v[i][1];//Left
        v[i][no_x+1]=-v[i][no_x];//Right
    }
    for(j=0;j<no_x+1;++j)
    {
        u[0][j]=-u[1][j];//Bottom
        u[no_y+1][j]=2-u[no_y][j];//Top
    }
    //Calculation of properties at nodes
    for(i=no_y;i>=0;--i)
    {
        for(j=1;j<no_x+2;++j)
            pr<<(p[i][j]+p[i+1][j]+p[i+1][j-1]+p[i][j-1])/4<<" ";
        pr<<"\n";
    }
    pr.close();
    //x-velocity
    for(i=no_y;i>=0;--i)
    {
        for(j=0;j<no_x+1;++j)
            x_vel<<(u[i][j]+u[i+1][j])/2<<" ";
        x_vel<<"\n";
    }
    x_vel.close();
    //y-velocity
    for(i=no_y;i>=0;--i)
    {
        for(j=0;j<no_x+1;++j)
            y_vel<<(v[i][j]+v[i][j+1])/2<<" ";
        y_vel<<"\n";
    }
    y_vel.close();
    //For contour and streamlines
```

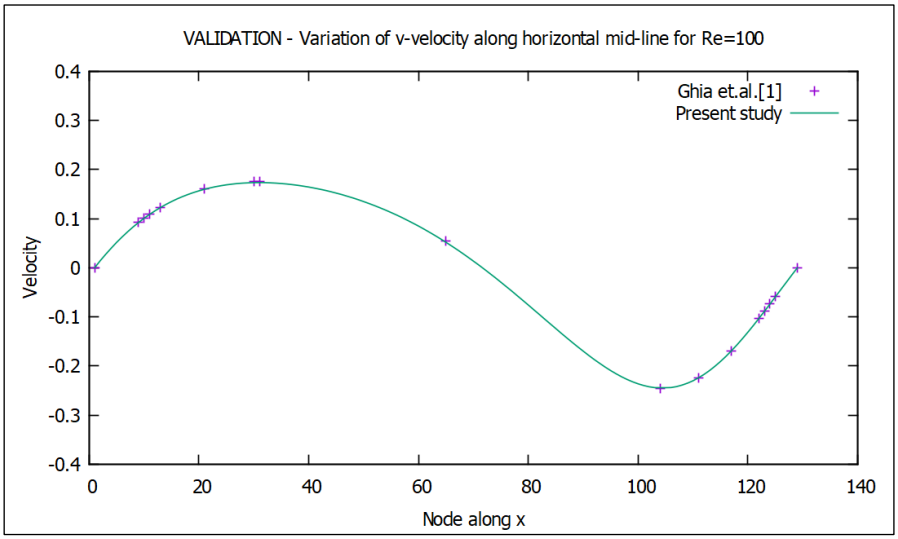
```
186       for(i=0;i<no_y+1;++i)
187       {
188           for(j=0;j<no_x+1;++j)
189           {
190               data<<j/float(no_x)<<" "<<i/float(no_y)<<" ";
191               data<<(p[i][j]+p[i+1][j]+p[i+1][j+1]+p[i][j+1])/4<<" ";
192               data<<(u[i][j]+u[i+1][j])/2<<" ";
193               data<<(v[i][j]+v[i][j+1])/2<<" ";
194
data<<sqrt(pow((u[i][j]+u[i+1][j])/2,2)+pow((v[i][j]+v[i+1][j])/2,2))<<"\n";
195           }
196       }
197       data.close();
198   }
```

Output

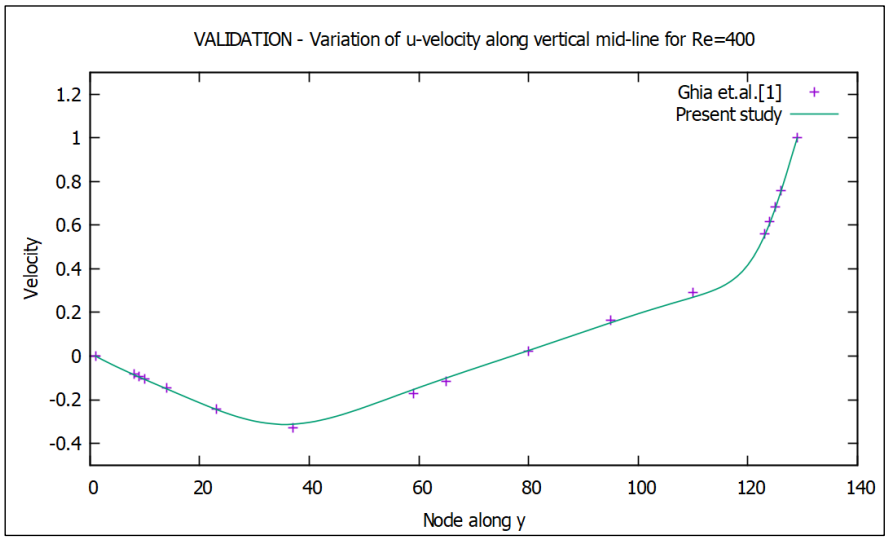
Validation

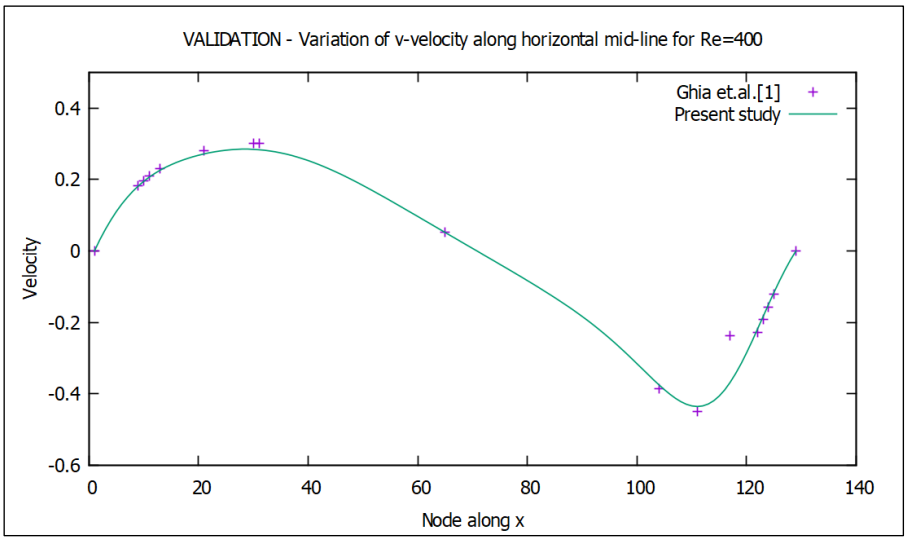
- *Re = 100*



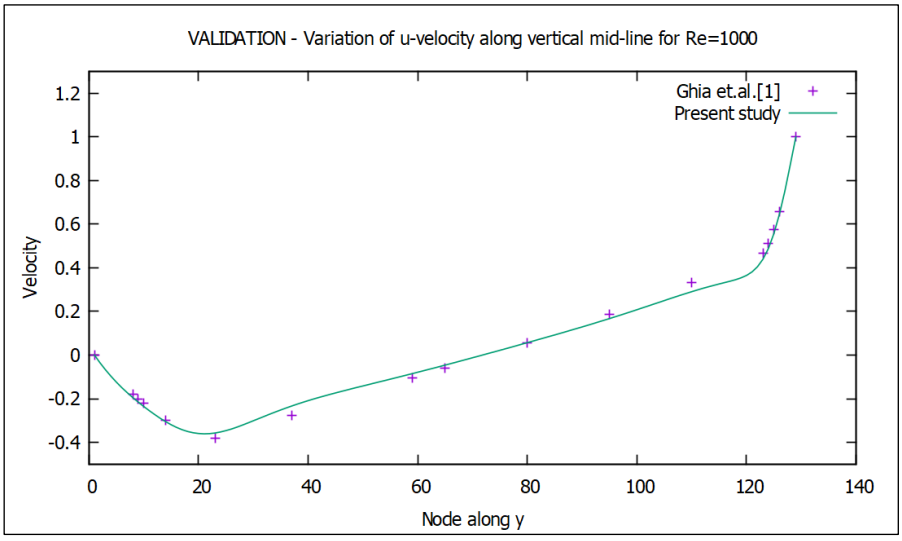


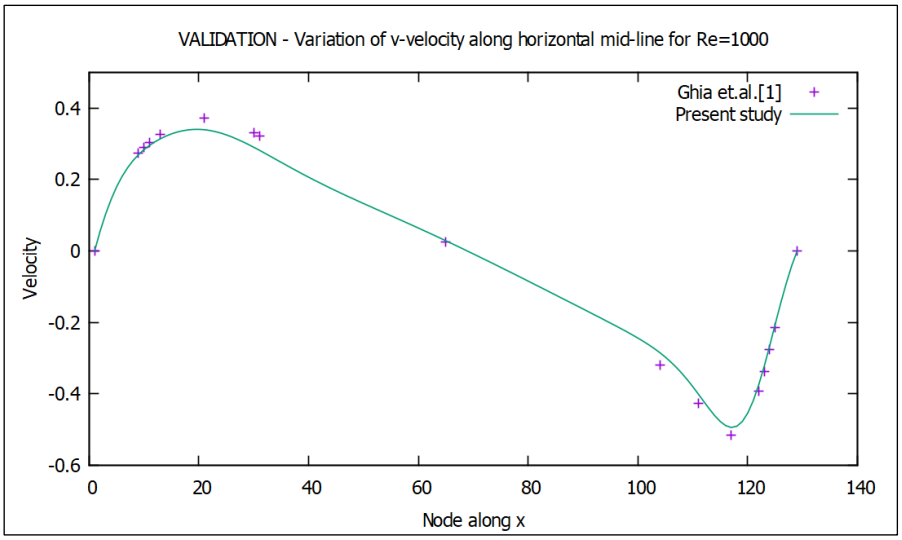
- *Re = 400*





- *Re = 1000*

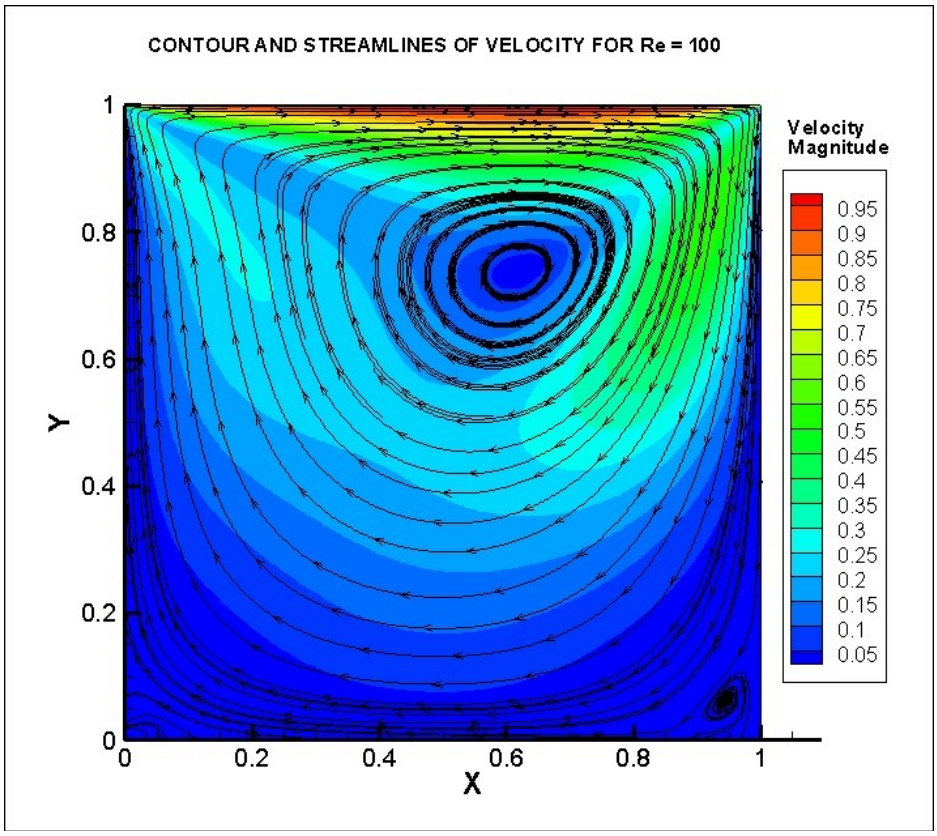




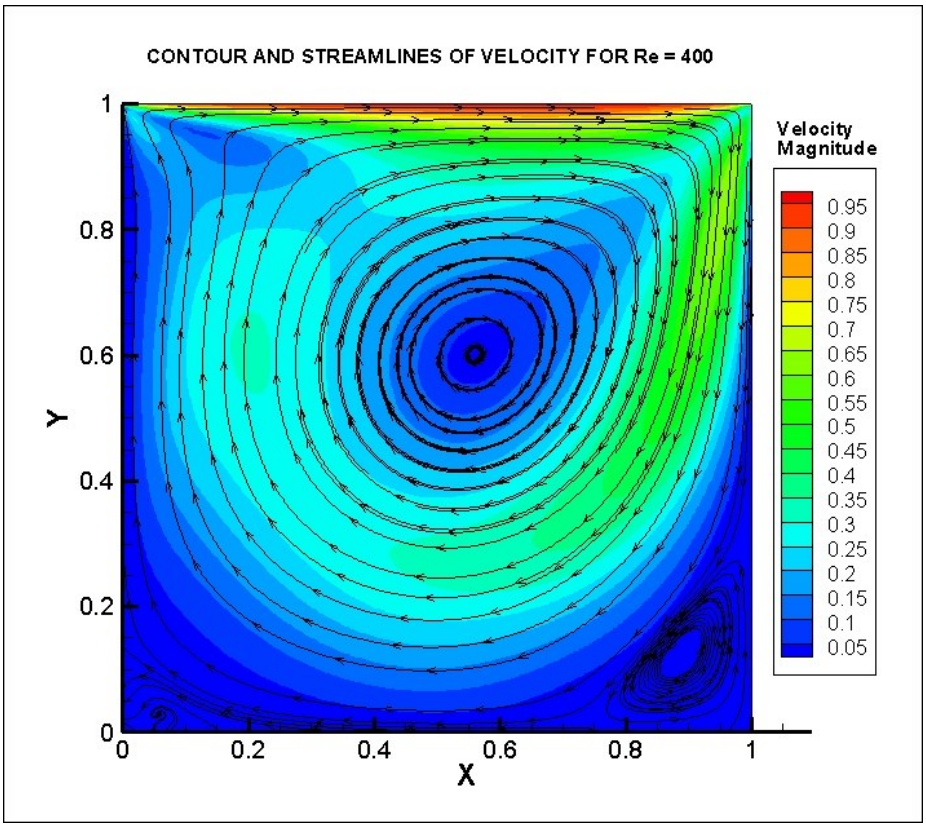
Observation: The present study seems to agree well with the well established results for lid driven cavity. Hence, it is accurate and further analyses can be done using the developed code.

Velocity contour and Streamline

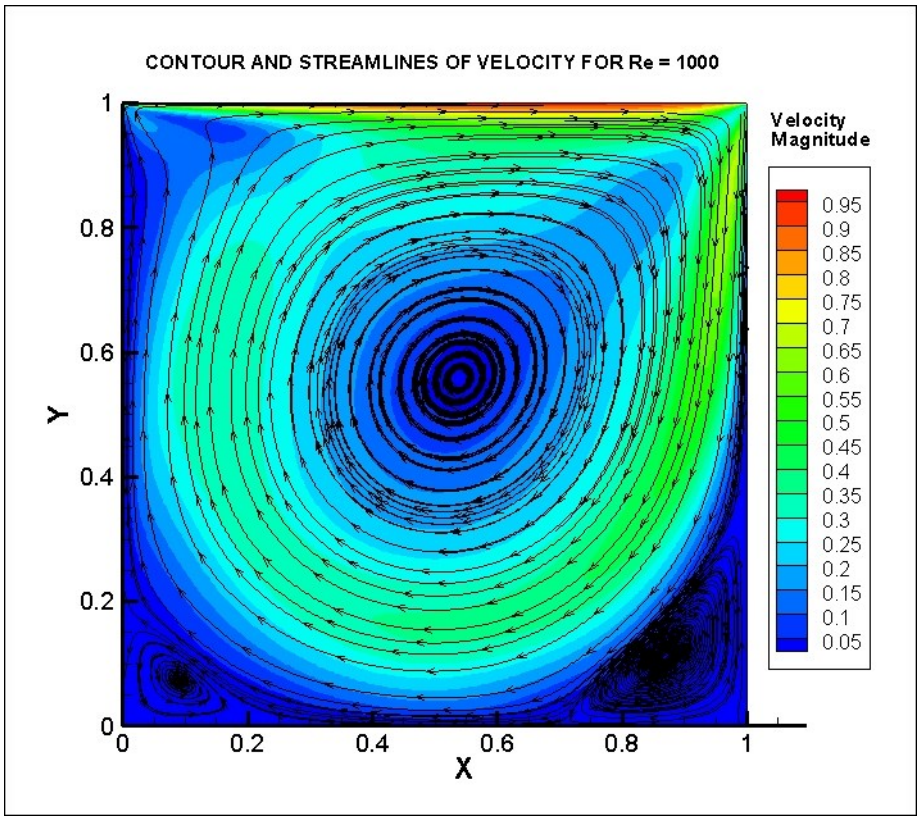
- *Re = 100*



- *Re = 400*

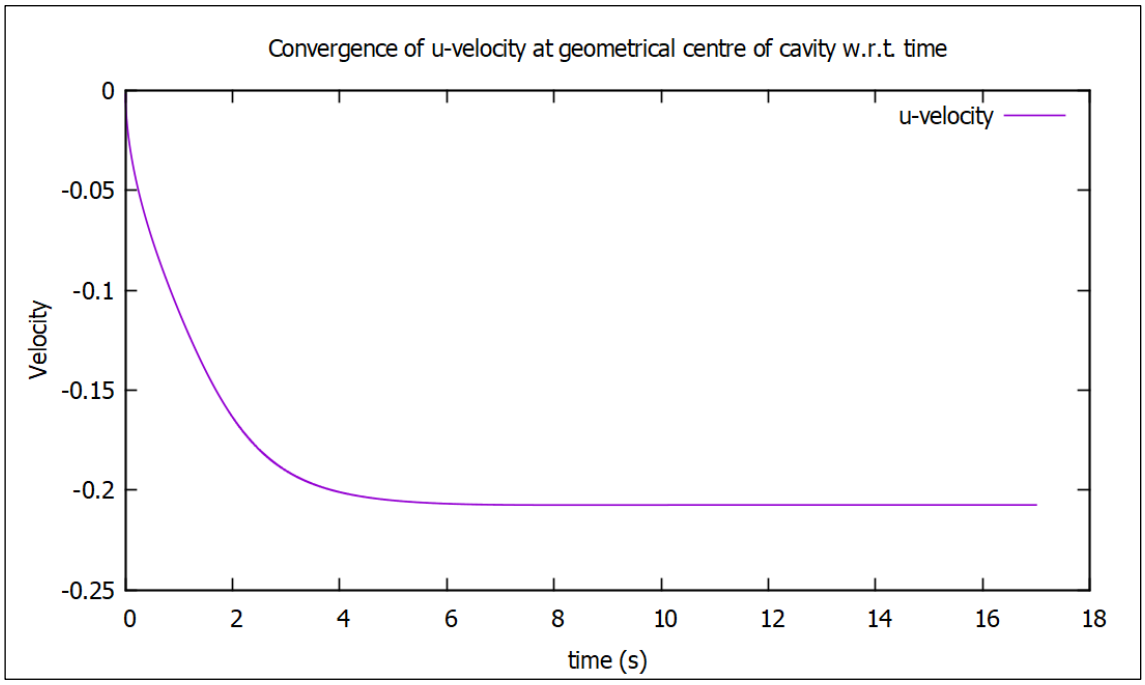


- *Re = 1000*



Observation: Clockwise circulation is observed within the cavity. Vortices at bottom edges become more significant with increse with increase in Reynolds number.

Convergence Plot



Reference:

[1] U Ghia, K.N Ghia, C.T Shin,High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method, Journal of Computational Physics, Volume 48, Issue 3, 1982, Pages 387-411, ISSN 0021-9991, https://doi.org/10.1016/0021-9991(82)90058-4.