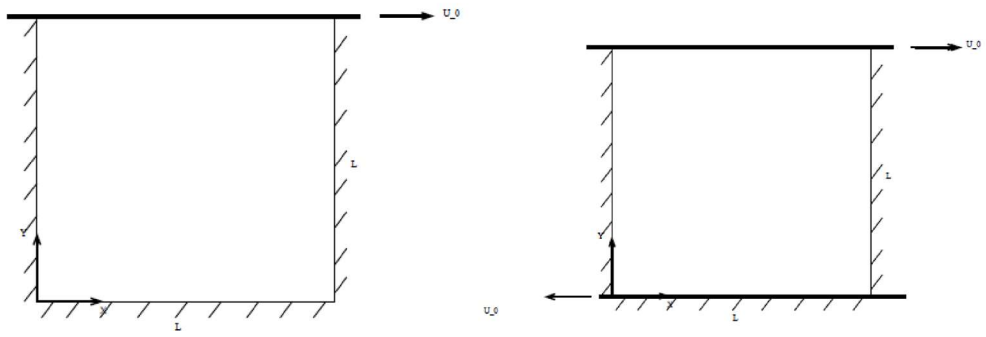
N-S Solver for 2-D lid driven cavity flow

Summary: Solve the incompressible Navier-Stokes equations in primitive variable form for the two-dimensional cavity problem.



Problem description：

Non-dimensionalize all the variables and governing equations (including initial and boundary conditions) using the length of the cavity (L) and the velocity of the plate (U_0) as scales for the length and velocity respectively. Derive the boundary condition for the pressure and velocity along the four walls of the cavity.

Validate the unsteady MAC code for a square, one sided lid-driven cavity flow at Re = 400. Investigate accuracy and stability by considering the effect of the time step dt and the grid spacing dx = dy. Compare the unsteady and steady results from literature.

Compute a case with different aspect ratio and investigate the effect of Reynolds number in the range Re = 1 to Re = 400.

Description of Methodology [1]:

First let's non-dimensionalize the equations first, the 2D laminar flow equations:

$$\frac{\partial u^*}{\partial x^*} + \frac{\partial v^*}{\partial y^*} = 0$$

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

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

Substitute $x = x^*/L$, $y = y^*/L$, $u = u^*/U_0$, $v = v^*/U_0$ $t = t^* U_0/L$ and $P = p^*/\rho U_0^2$

into the original equations, non-dimensional form is obtained:

$$\begin{cases} \dfrac{\partial u}{\partial x}+\dfrac{\partial v}{\partial y}=0.....(1) \\ \dfrac{\partial u}{\partial t}+\dfrac{\partial uu}{\partial x}+\dfrac{\partial uv}{\partial y}+\dfrac{\partial p}{\partial x}=\dfrac{1}{\mathrm{Re}}(\dfrac{\partial^2 u}{\partial x^2}+\dfrac{\partial^2 u}{\partial y^2}).....(2) \\ \dfrac{\partial v}{\partial t}+\dfrac{\partial vv}{\partial y}+\dfrac{\partial uv}{\partial x}+\dfrac{\partial p}{\partial y}=\dfrac{1}{\mathrm{Re}}(\dfrac{\partial^2 v}{\partial x^2}+\dfrac{\partial^2 v}{\partial y^2}).....(3) \end{cases}$$

Discretize the equations in FTCS (Forward in time and Central difference in space)

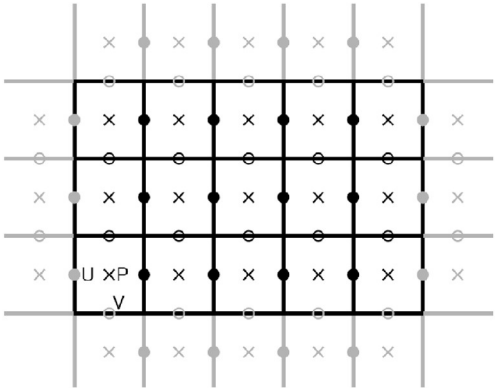
For equation (2): $\left.\dfrac{\partial u}{\partial t}\right|_{i+\frac{1}{2},j}^{n}=\dfrac{u_{i+\frac{1}{2},j}^{n+1}-u_{i+\frac{1}{2},j}^{n}}{\Delta t}+o(\Delta t)$, $\left.\dfrac{\partial p}{\partial x}\right|_{i+\frac{1}{2},j}^{n}=\dfrac{p_{i+1,j}^{n}-p_{i,j}^{n}}{\Delta x}+o(\Delta x^2)$,

$\left.\dfrac{\partial^2 u}{\partial x^2}\right|_{i+\frac{1}{2},j}=\dfrac{u_{i+\frac{3}{2},j}-u_{i+\frac{1}{2},j}+u_{i-\frac{1}{2},j}}{\Delta x}+o(\Delta x^2)$, $\left.\dfrac{\partial^2 u}{\partial y^2}\right|_{i+\frac{1}{2},j}=\dfrac{u_{i+\frac{1}{2},j+1}-u_{i+\frac{1}{2},j}+u_{i+\frac{1}{2},j-1}}{\Delta y^2}+o(\Delta y^2)$,

$\left.\dfrac{\partial uu}{\partial x}\right|_{i+\frac{1}{2},j}=\dfrac{(u_{i+1,j})^2-(u_{i,j})^2}{\Delta x}+o(\Delta x^2)$, $\left.\dfrac{\partial uv}{\partial y}\right|_{i+\frac{1}{2},j}=\dfrac{(uv)_{i+\frac{1}{2},j+\frac{1}{2}}-(uv)_{i+\frac{1}{2},j-\frac{1}{2}}}{\Delta y}+o(\Delta y^2)$ (in which $(uv)_{i+\frac{1}{2},j+\frac{1}{2}}=(\dfrac{u_{i+\frac{1}{2},j}+u_{i+\frac{1}{2},j+1}}{2})(\dfrac{v_{i+1,j+\frac{1}{2}}+v_{i,j+\frac{1}{2}}}{2})$) Similar for equation (3).

And for the continuity equation (1): $\dfrac{\partial u}{\partial x}+\dfrac{\partial v}{\partial y}=\dfrac{u_{i+\frac{1}{2},j}-u_{i-\frac{1}{2},j}}{\Delta x}+o(\Delta x^2)+\dfrac{u_{i,j+\frac{1}{2}}-u_{i,j-\frac{1}{2}}}{\Delta y}+o(\Delta y^2)=0$

Discretize the equation on a staggered grid (Mark and Cell method [6])



Numerically solving the Navier-Stokes equation on the staggered grid

1) Get a temporary velocity with

$$\frac{u^* - u^n}{\Delta t} = -(convection\ terms) + (diffusion\ terms) \qquad \frac{v^* - v^n}{\Delta t} = -(convection\ terms) + (diffusion\ terms)$$

$$u^* = u^n + \Delta t(-convection\ terms + diffusion\ terms) \qquad v^* = v^n + \Delta t(-convection\ terms + diffusion\ terms)$$

2) Obtain the pressure Poisson equation by using the temporary velocity

$\frac{-\nabla \cdot (u^*)}{\Delta t} = -\nabla^2 p$, $-\left(\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2}\right) = -\left(\frac{1}{\Delta t}\right)\left(\frac{\partial u^*}{\partial x} + \frac{\partial v^*}{\partial y}\right)$

in which LHS ,

and RHS $-\left(\frac{1}{\Delta t}\right)\left(\frac{\partial u^*}{\partial x} + \frac{\partial v^*}{\partial y}\right) = -\left(\frac{1}{\Delta t}\right)\left(\frac{u^*_{i,j} - u^*_{i-1,j}}{\Delta x} + \frac{v^*_{i,j} - v^*_{i,j-1}}{\Delta y}\right)$

3) Solve the pressure Poisson equation iteratively. In our case, instead of directly applying Gauss-Seidel iteration, a successive over relaxation method is used and a relaxation factor rf = 1.6 is used to speed up the convergence speed (N steps instead of N^2), and the relaxation factor would be the best option if it can minimize the spectral radius of the iteration matrix. In this code a constant relaxation factor rf = 1.6 is chosen for simplicity's sake.

The pressure boundary condition (artificial pressure on the ghost points):

$$p_{i,1} = p_{i,2} - 2\frac{v_{i,\frac{5}{2}}}{\mathrm{Re}\,\Delta y} \qquad p_{i,n+2} = p_{i,n+1} - 2\frac{v_{i,n+\frac{1}{2}}}{\mathrm{Re}\,\Delta y} \qquad p_{1,i} = p_{2,i} - 2\frac{u_{\frac{5}{2},i}}{\mathrm{Re}\,\Delta x} \qquad p_{n+2,1} = p_{n+1,i} + 2\frac{u_{i,n+\frac{1}{2}}}{\mathrm{Re}\,\Delta x}$$

(P is an (Nx+2) x(Ny+2) matrix, u is (Nx+1)x Ny and v is (Nx)x (Ny+1), all including ghost points)

4) Corrector step, of the projection method is applied as shown below to obtain the correct velocity field: $u_{ij}^{n+1} = u_{ij}^* - \Delta t\left(\frac{p_{i+1,j} - p_{i,j}}{\Delta x}\right)$ $v_{ij}^{n+1} = v_{ij}^* - \Delta t\left(\frac{p_{i,j+1} - p_{i,j}}{\Delta y}\right)$. Plot results at every certain time.

5) repeat 1) to 4) until final time (or a steady state)

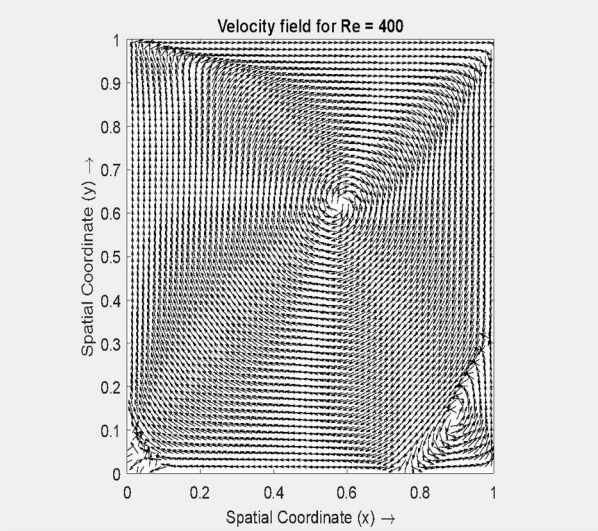
6) reduce dt or dx by a certain integer number and study the convergence rate.

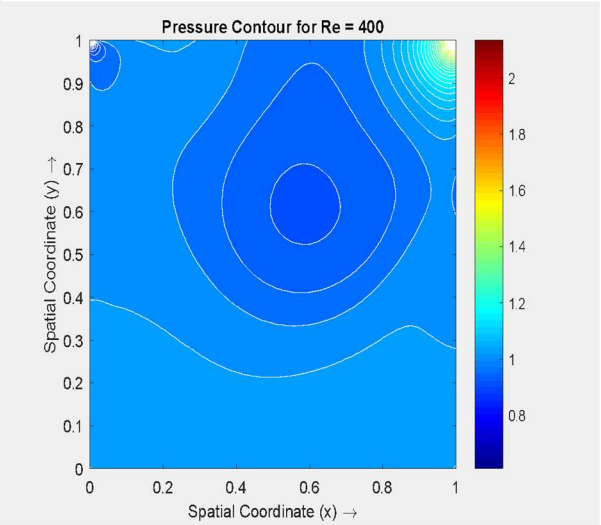
7) Change the bottom velocity/aspect ratio, calculate results and compare to the literatures.

Results and discussion

For Re = 400

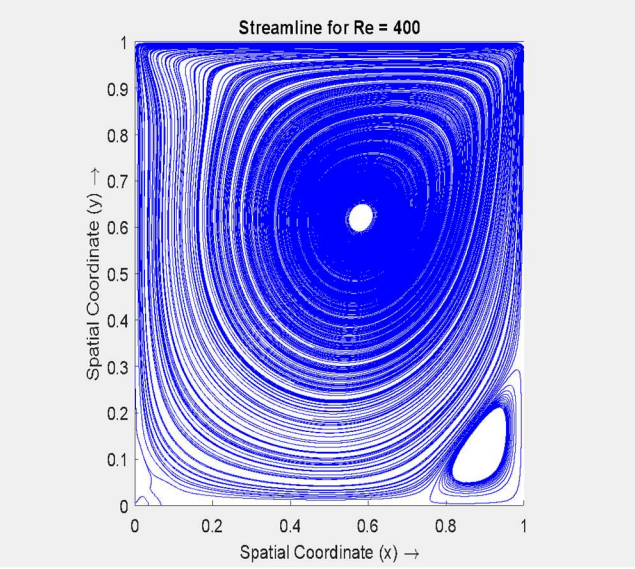
For a calculation with 128x128 grid, dt = 0.001. The velocity field and streamline plot:

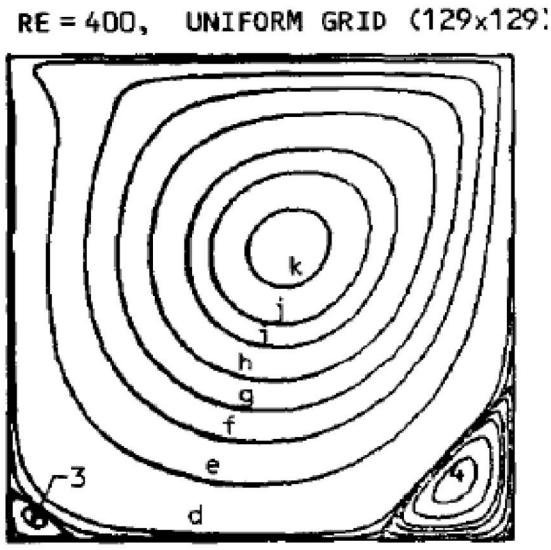




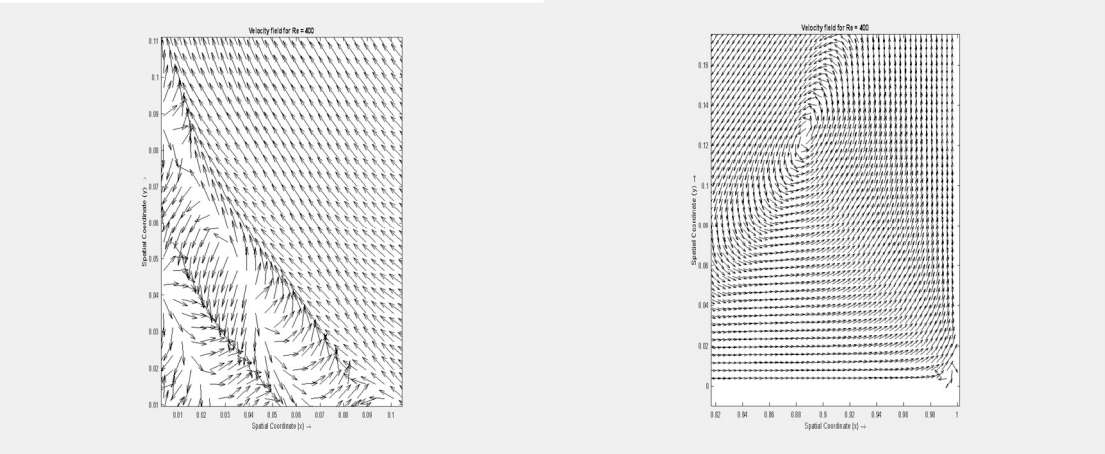
When the flow is developing, some “oscillation” are found in the velocity field at the left down corner.

And the streamline plot, the one on the right side is the streamline plot in the literature (Ghia, 1982[2])





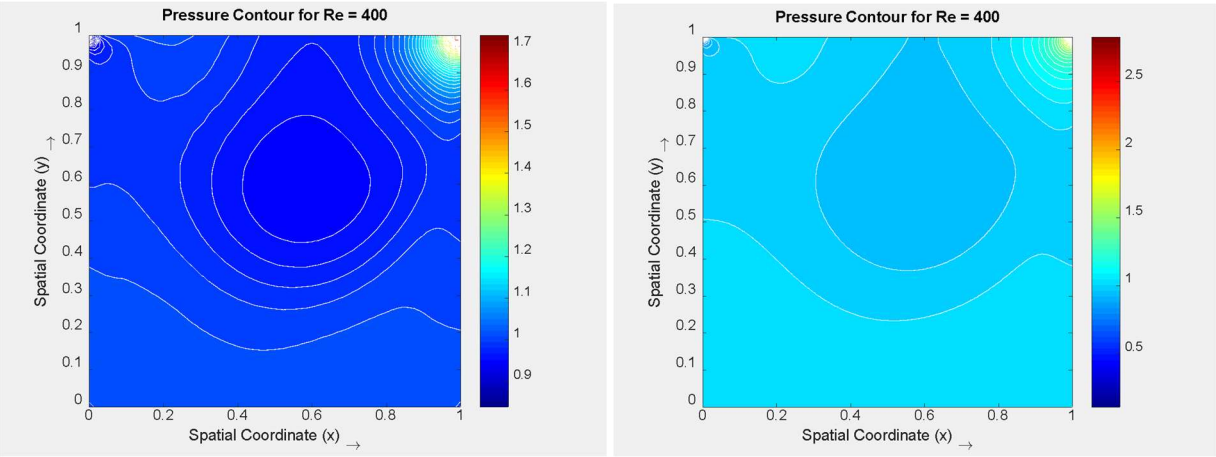
For a calculation with 256x256 grid, dt = 0.001. The velocity field at the corner:



The left-down corner (left) and right-down (right) corner of the velocity field plot are shown as above. Vortices break down into smaller ones at the corner, until the scale is small enough that viscosity comes to dominant. We can see as the previous "oscillation" at the left-down corner seems to be many small vortices, which are not well resolved for the previous coarse grid. However, it might not be necessary to resolve every single small vortex forming in the flow. An average value of a small region may be sufficient for engineering purpose.

Note that for the same grid number, the results at left down corner isn't as clear as the results from literature (which can be seen in the velocity field and streamline plot), and can hardly reach steady state due to oscillation. This may due to the numerical diffusion caused by the scheme we used.

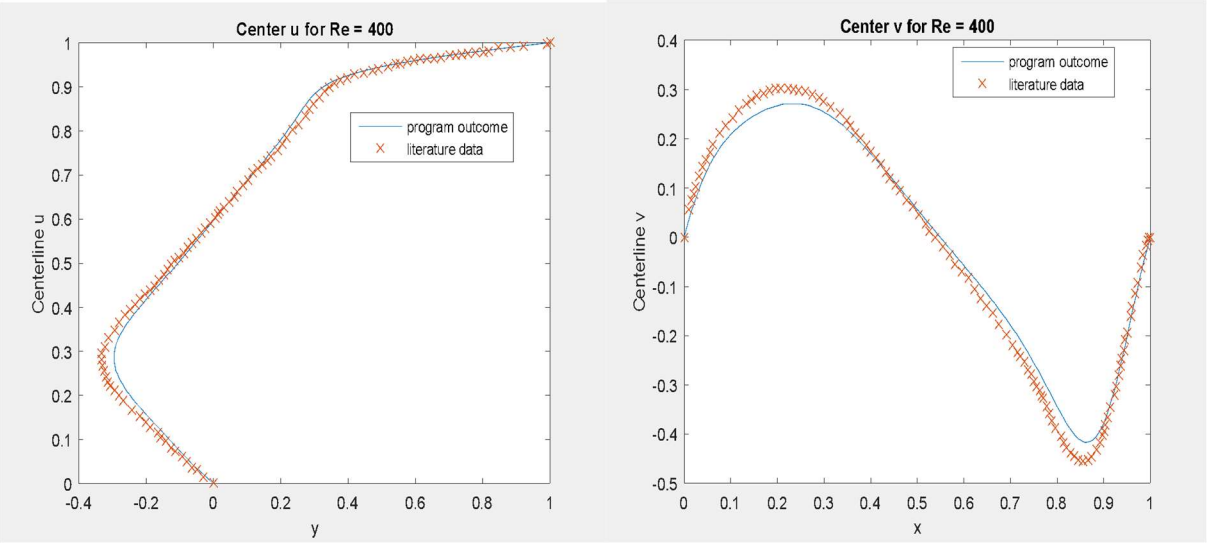
About numerical diffusion in the solver, we can compare the pressure contour from different grid spacing. The pressure contour from 64x64 grid (left) and 256x256 grid (right) are shown as below:



Comparing to the pressure contour from 128 x 128 grid, we notice that the maximum value at the upper left corner (which is supposed to be a pressure singularity point) increase as we refine the grid. This is because the fact the central difference scheme in space is numerically dissipative, introducing an artificial diffusivity, which smear out large spatial gradience (instead of having a sharp interface over 1 cell, the space discretization operator will spread it over a few cells). As the grid spacing gets smaller and smaller, the artificial diffusivity is diminishing and we start to see that singularity point more and more clearly.

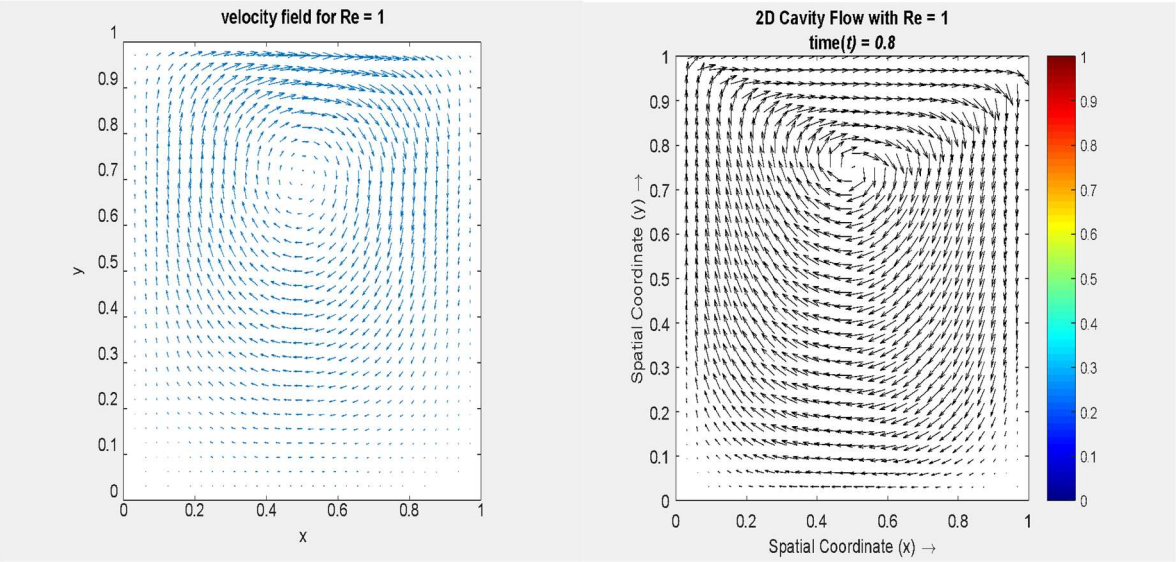
Adding a group of grid points with constant velocity u = 1 on the top as “main stream”, or simply using a modified top velocity profile instead of u = 1, could help eliminate the singularity points.

Comparing the centerline velocity with data from literature (Marchi, C.H, 2009 [3])



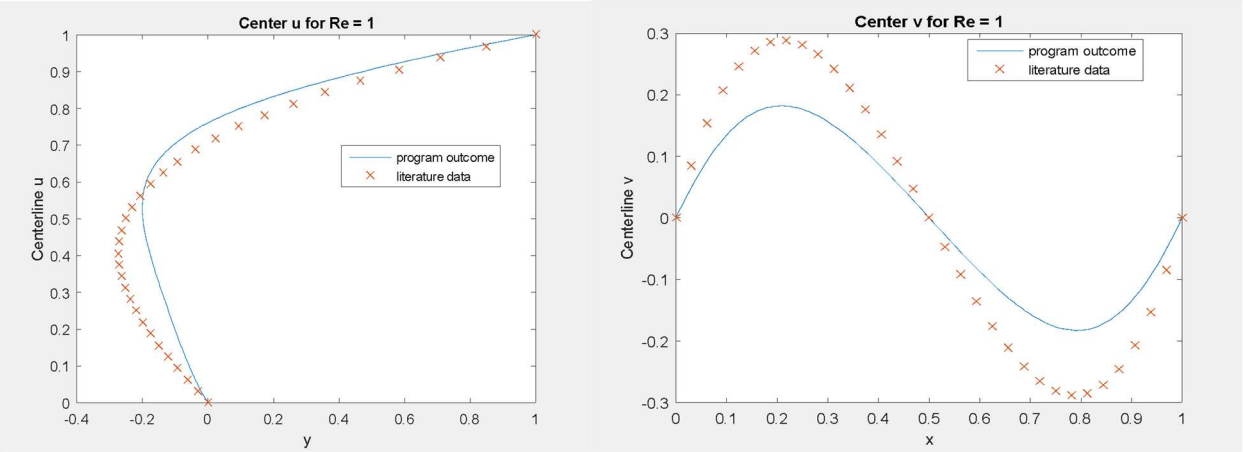
The centerline u and v velocity shows a good match between the literature data and program outcome, except some certain values seem to be off a bit. This may be due to the computation limit 30 years ago for the literature data.

Re = 1



A major drawback of FTCS scheme (the scheme required to use in this problem) is for problems with large diffusivity (1/Re in our case) the time step restriction can be too severe. For Re = 1, the time step need to be $\Delta t < 0.25\Delta x^2$ to get a stable solution, for example, $\Delta t < 0.000061$ for a coarse grid like 64x64. Even though implicit Euler scheme (an unfinished work for this project) for time requires to invert

an $N^2 \times N^2$ matrix for an $N \times N$ grid, it would not be a big issue to our problem, considering the finest grid in this case is 1024x1024, which doesn't take that much space to store the operator matrix. Furthermore, MATLAB comes with many decent in-built function for matrix like LU decomposition lu , Cholesky decomposition chol and mldivide \. They allow us to go through the calculation without the pain of directly invert the matrix, and get a more compact code without those inefficiency "for" loops.



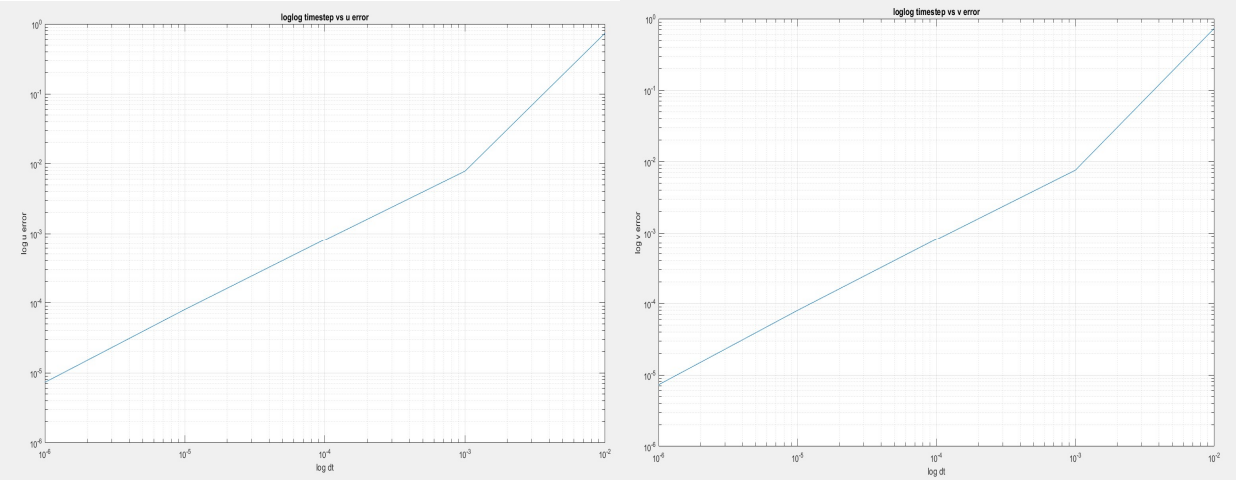
Comparing the centerline velocity between code outcomes and the analytical solution for main vortex.

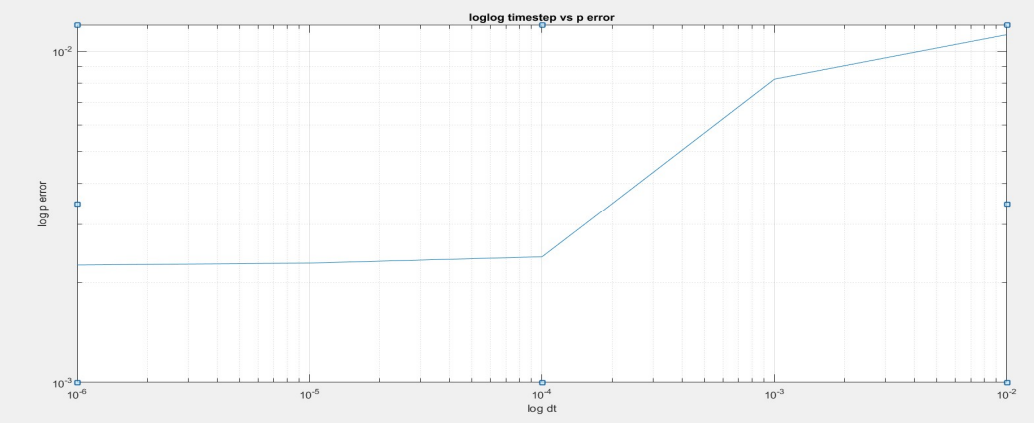
Noted that the analytical solution (Re = 1) modified the top velocity profile from u = 1 to $U_{top} = 16(x^4 - 2x^3 + x^2)$ in order to eliminate the singularity points at the two upper corners, so the difference in velocity magnitude is acceptable.

Stability requirement：CFL $0.25\,(|u|+|v|)^2\,\Delta t_1\,Re \leqslant 1$ and $\dfrac{\Delta t_2}{Re\,(\Delta x)^2} \leqslant 0.25$. $\Delta t \leqslant Min(\Delta t_1, \Delta t_2)$

Convergence study (Comparing the flow at t = 0.1 with different time step and grid spacing)

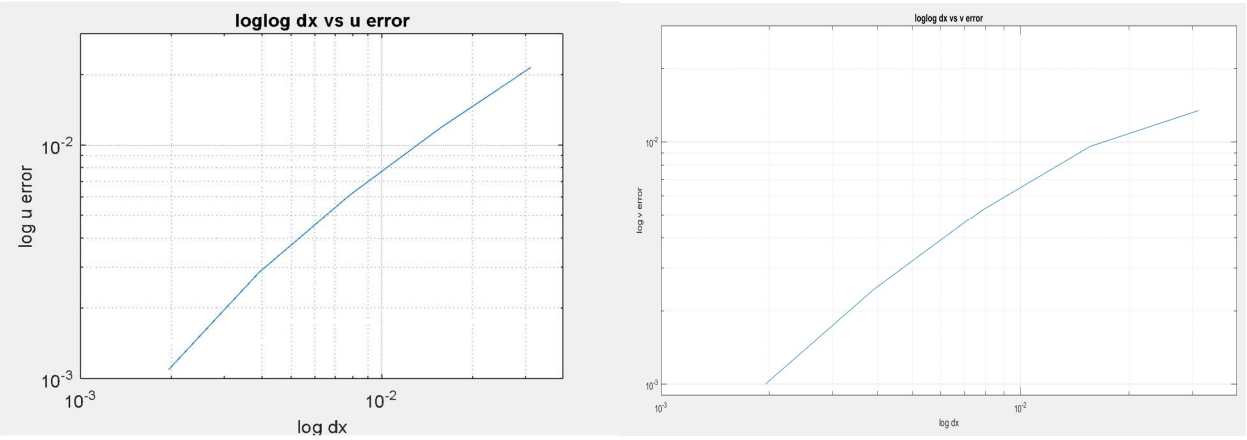
For Re = 400, square cavity, one side driven (on the top) case

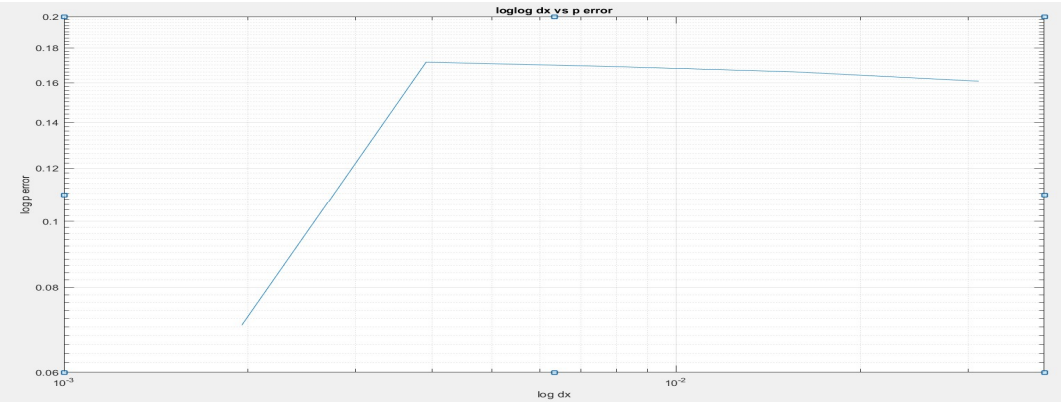




$$\left.\frac{\partial u}{\partial t}\right|^{n}_{i+\frac{1}{2},j} = \frac{u^{n+1}_{i+\frac{1}{2},j} - u^{n}_{i+\frac{1}{2},j}}{\Delta t} + o(\Delta t)$$

(first order forward Euler scheme). The data for dt = 0.01 is outside the asymptotic range (it doesn't meet the stability criteria d, and wouldn't get a stable solution for final time > 0.1. The slope for the log-log plot of dt vs velocity from dt = 1e-3 to dt = 1e-6 matches the error order guess well.

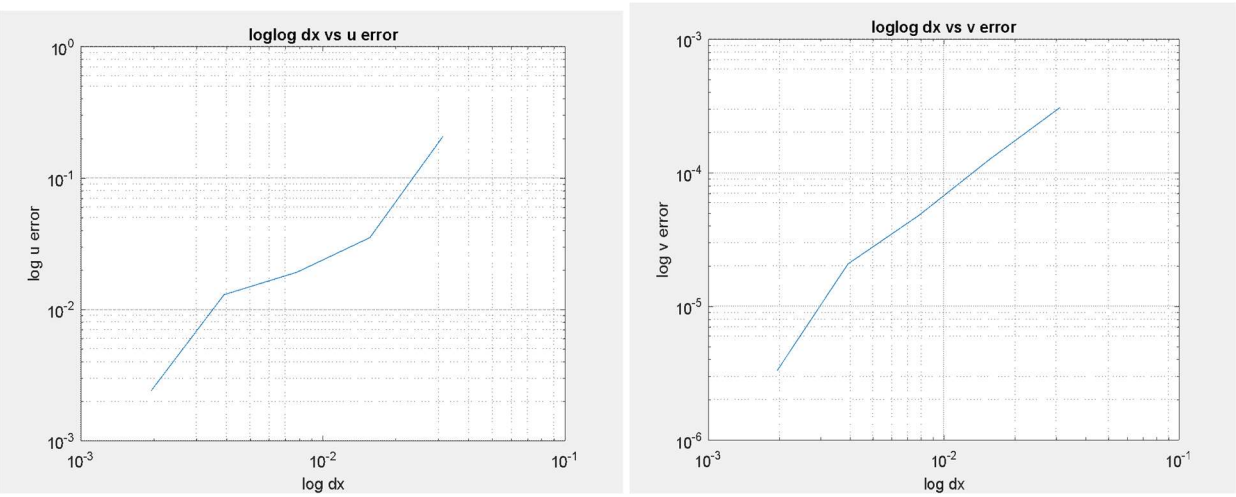




Central difference scheme was applied to spatial discretization. The slope for the log-log plot of dx vs velocity from dx = 1/32 to dx = 1/1024 indicates a truncation error on the order of O(dx) instead of O(dx^2). This might because that at the boundaries we use a second order accurate scheme to account for

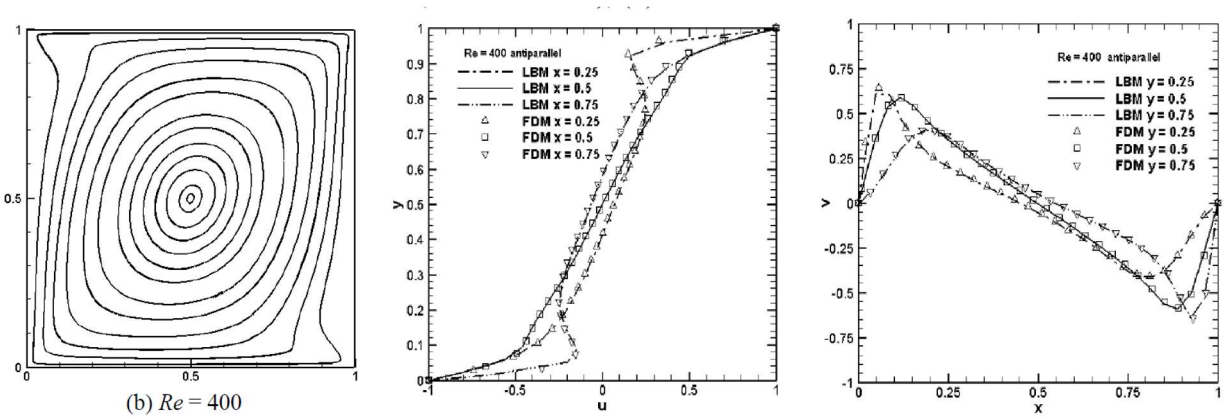
the boundary conditions that is next involved in the second order derivatives (Laplacian). It results in a truncation error O(1) at the boundaries (U_boundary comes with a truncation error on the order of o(dx^2), then we do central difference with that value to get the discretized form equations).

If we take a certain point within the domain for example, point ($0.5L_x$, $0.95L_y$), instead of doing root mean square difference for the data of entire domain, it is possible that the error order for u and v would be closer to o(dx^2) rather than o(dx).



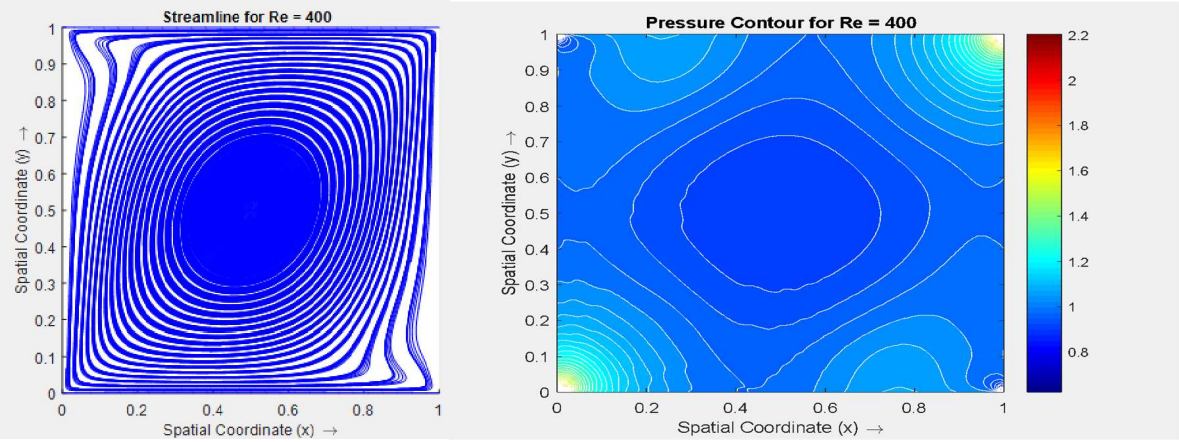
The log-log plot of dx vs error for point ($0.5L_x$, $0.95L_y$), we can look at the slopes, $q_1 \approx 1.60$ for u error log-log plot and $q_2 \approx 1.63$ for v, which are closer to a second-order truncation error.

Two side driven cavity flow (Re = 400):

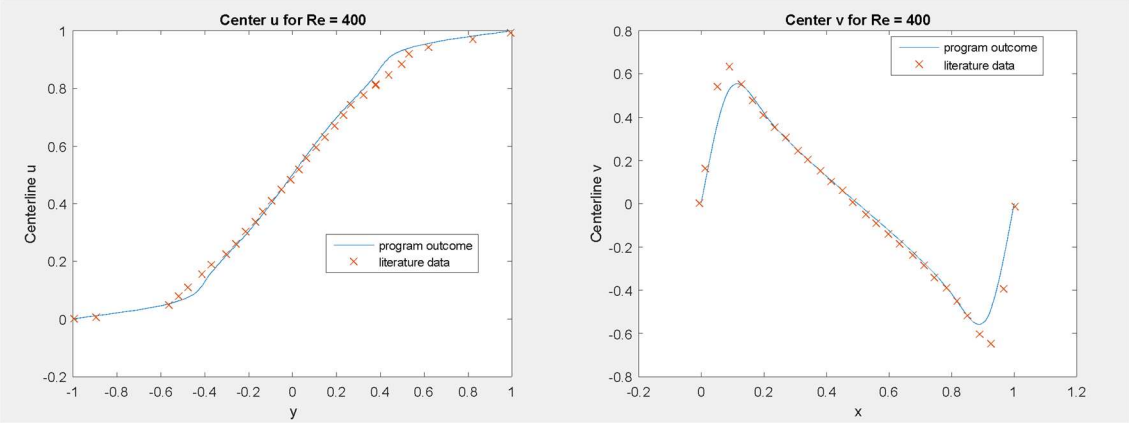


(b) *Re* = 400

Literature results (streamline and center velocity profile) of anti-parallel wall motion cavity(bottom and left, aspect ratio = 1, given by Lattice Boltzmann method[4]).



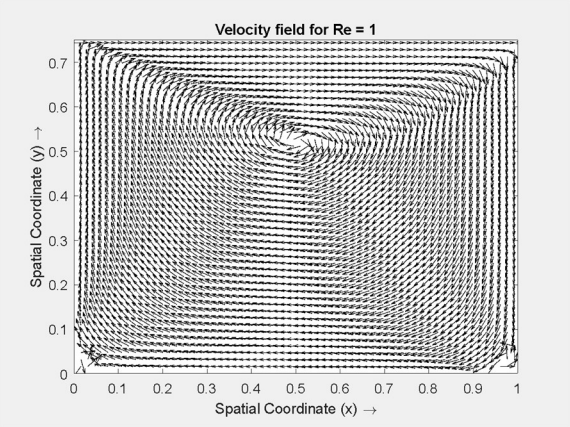
Streamline pattern of the program calculation result

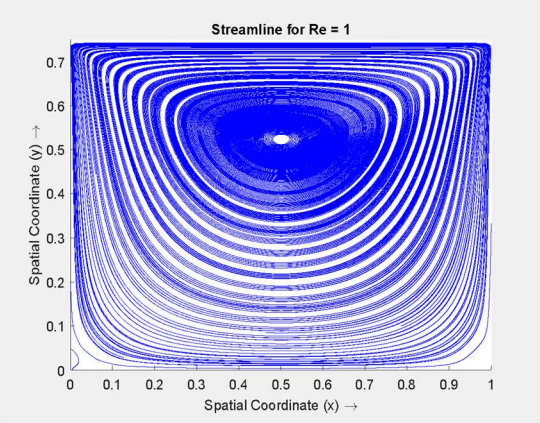


`Center u(left) and v(right); calculation results vs literature data (` [4] Perumal, & Dass., 2010)

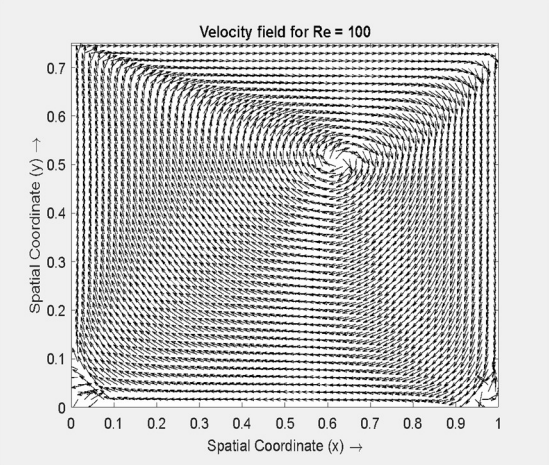
Centerline u and v velocity plots show decent fits with the literature data. The calculation outcomes mostly match the literature data well.

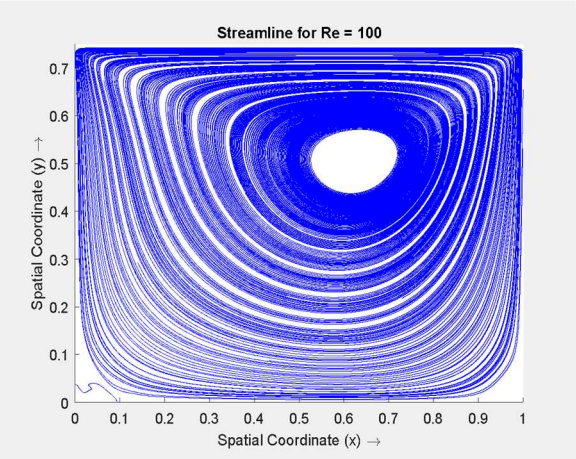
Aspect ratio: Study the effect of Reynolds Number on cavity flow with aspect ratio A = 4/3



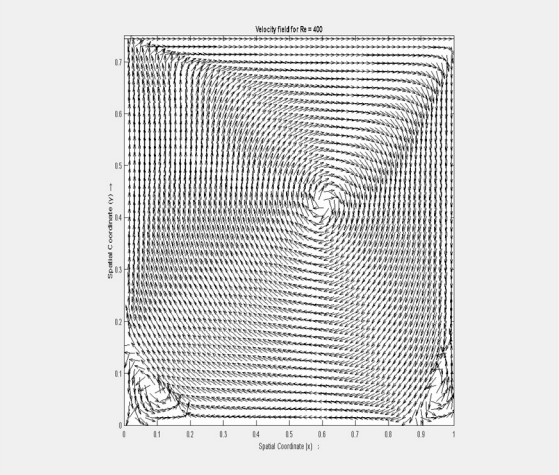


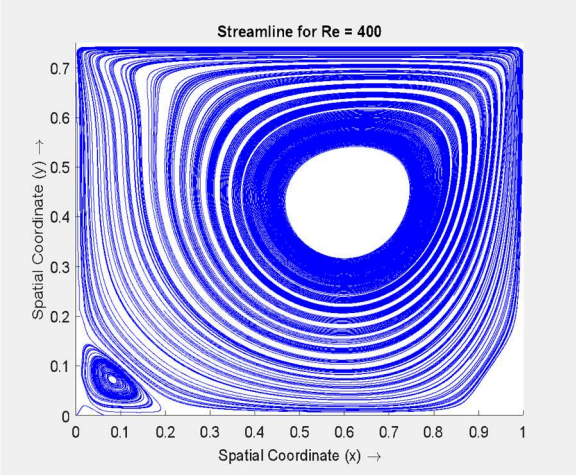
Re = 400 (128 x 128 grid, dt = 0.00001)

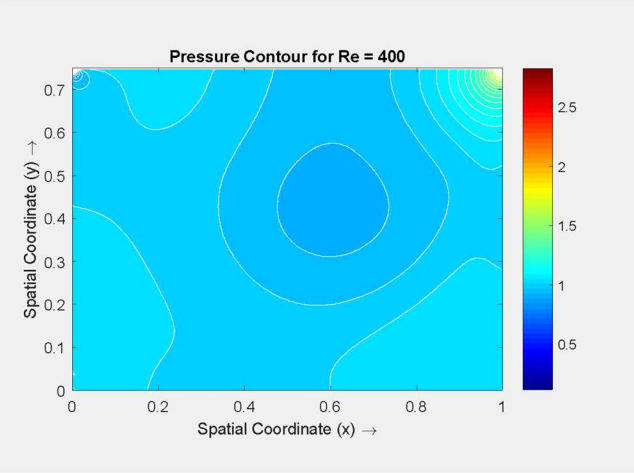




Re = 400 (128 x 128 grid dt = 0.005)

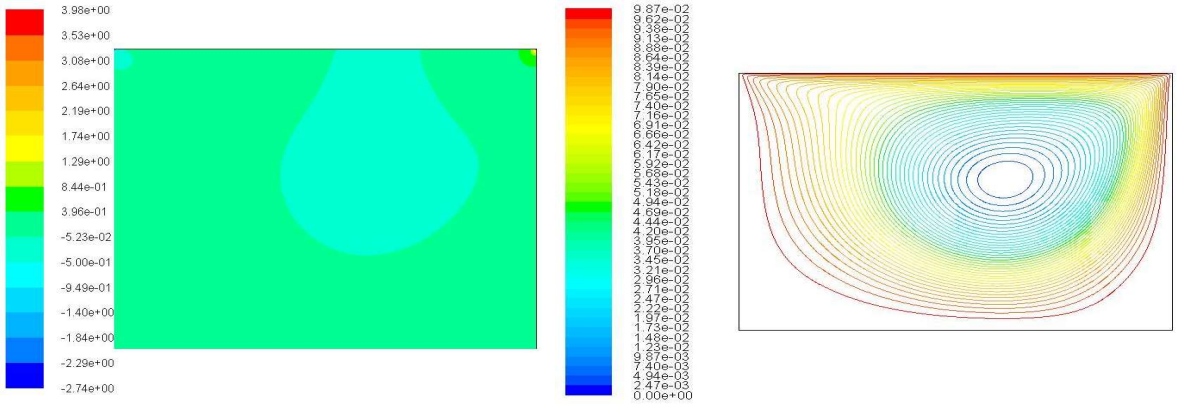






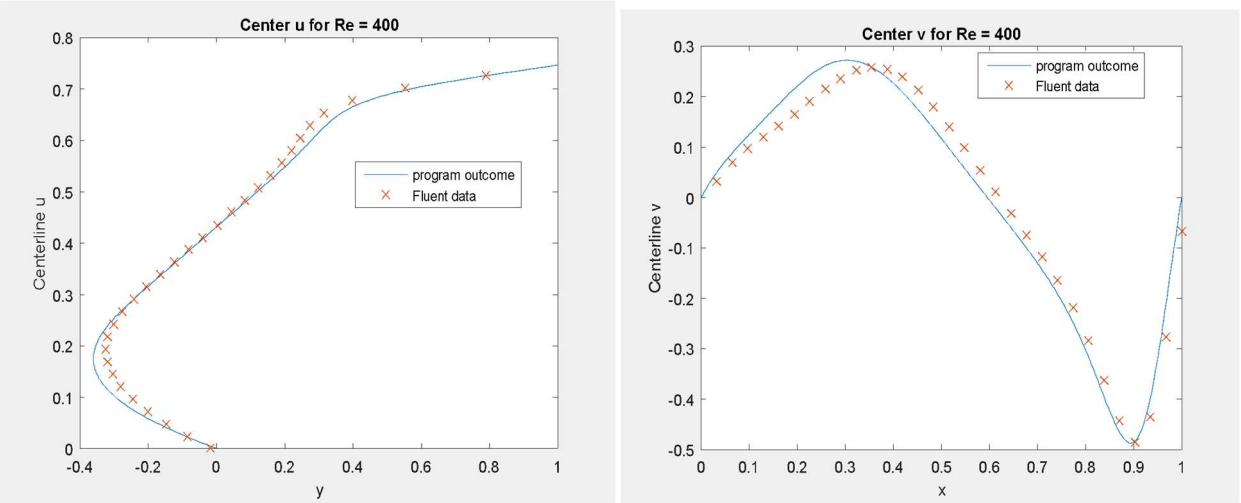
Velocity field, streamline plot and pressure contour for Re = 400 (128 x 128 grid dt = 0.005)

Literature results for aspect ratio A = 4/3 are not found, so we compare the Re = 400 case to an ANSYS Fluent simulation result.



Pressure contour (left) and streamline plot (right)

Centerline u plot(left) and v plot(right) from fluent data



The calculation results match the Fluent simulation results well.

The Fluent set up: fluid density $\rho = 1$, "viscosity", $\eta = \frac{1}{\mathrm{Re}} = 0.0025$, SI unit (which should be non-dimensional parameters). Solution method: Simple; Gradient: Least Squares Cell Based;

Pressure: Standard; Momentum-Second order upwind; Residuals: 1e-9

We can see from the plots at different Reynolds number: as we chose dt < dt (CFL condition), the scheme well capture the nonlinear term behavior at high Reynold number, the streamlines become less and less symmetric because of the convective terms that come into play.

Since the aspect ratio A is still relatively small, the flow is still open type cavity flow and far from even becoming transitional type, but for higher Reynolds number like Re = 400, the main vortex becomes less "dominant" than the square cavity case, and left down corner vortex plays a more important role in the flow pattern. For an aspect ratio A = 4/3, It would also require more grid points not only to resolve the small corner vortices, but also to resolve the "shear layer" on the top edge, which becomes thinner and thinner at high Reynolds number.

Conclusion

Through the most basic scheme and a given iterative solver for pressure Poisson equations, the Navier - Stoke could simulate the 2-D laminar flow in lid-driven cavity. The results for square cavity at Re = 400 match literature results well mostly, however the FTCS scheme left much room to improve computation performance, given the fact that the scheme puts heavy restriction on the time step due to stability issues.

The scheme used in the program gives a time discretization error on the order of $o(\Delta t)$, and is supposed to have a spatial discretization error on the order of $o(\Delta x^2)$. But because of the error introduced when

interpolating for boundary conditions, the order of spatial discretization error is between $o(\Delta x)$ and $o(\Delta x^2)$. Numerical diffusion also affect the results' accuracy.

Furthermore, two side driven cavity flow and cavity flow with aspect ratio A=4/3 are also studies. The computation results are compared to the literature data or Fluent simulation results and they match well mostly.

Program Listing:

Navier-Stokes solver for cavity flow

```
function FTCS_MAC()
%%%%%%%%%% Setup %%%%%%%%%%
clc; close all; clear all;
% grid
Nc        = 128;          % cell number
Nx          = Nc + 2;     % cell number including ghost points
Ap = 0.75;                % aspect ratio
Lx = 1;    dx    = Lx/(Nx-1);
Ly = Lx*Ap;
dy = dx;
Ny = floor(Ly/dy)+1;
x  = 0:dx:Lx;                  y     = 0:dy:Ly;
% time step
dt        = 0.00001;                % timestep
Final_Time = 1;   % final time
dtout      = 0.01;
Nsteps     = floor(Final_Time/dt);
nout       = floor(dtout/dt);    % number of time steps for output
%parameters
Re         = 1;       % Reynolds number
nu         = 1/Re;
cfl        = 0.5;       % cfl criterion
% allocate variables
p = ones(Nx,Ny);             u = zeros(Nx+1,Ny);        v = zeros(Nx,Ny+1);
Q = zeros(Nx-2,Ny-2);        F = zeros(Nx+1,Ny);        G = zeros(Nx,Ny+1);
% velocity profiles
un = 1;     vn = 0;    us = 0;     vs = 0;
ue = 0;     ve = 0;    uw = 0;     vw = 0;
% boundary condtions %
  % u-vel
  u(1,:) = 2*uw - u(2,:); u(end,:)= 2*ue-u(end-1,:);   % Left and rightwall
  u(:,1) = us;            u(:,end) = un;  % Bottom and top wall
  % v-vel
  v(:,1) = 2*vs - v(:,2); v(:,end)= 2*vn - v(:,end-1);   % Bottom and top wall
  v(1,:) = vw;            v(end,:) = ve;   % Left and right wall
%%
%%%%%%%%%% main time loop %%%%%%%%%%
for tstep = 1:Nsteps
% check largest time step %
Umax = max(max(abs(u))); Vmax = max(max(abs(v)));
dt1 = 1/(Umax+Vmax)^2/cfl^2/Re;
dt2 = (0.25)*(dx^2)*Re;
dta = [dt1,dt2]; dt_max = min(dta);
if (dt > dt_max) , break;  end
%%%%%%%%%% compute f,g,q %%%%%%%%%%
```

```
    i = 2:Nx-2;
    j = 2:Ny-1;
       F(i+1,j) = u(i+1,j) + dt*((u(i+2,  j) - 2.*u(i+1,j) + u(i,  j)) * nu/ (dx^2)...
         + (u(i+1,  j-1) - 2.*u(i+1,j) + u(i+1,  j+1)) * nu / ( dy^2)...
         - ((0.5.*(u(i+1,j) + u(i+2,j)) ).^2 - (0.5.*(u(i,j) + u(i+1,j)) ).^2)/dx...
         - ((u(i+1,j)+u(i+1,j+1)).*(v(i+1,j+1)+v(i,j+1))...
         - (u(i+1,j)+u(i+1,j-1)).*(v(i+1,j)+v(i,j)))*0.25/dy );

   i = 2:Nx-1;
    j = 2:Ny-2;
       G(i,j+1) = v(i,j+1) + dt*((v(i+1,j+1) - 2*v(i,j+1) + v(i-1,j+1)) * nu/ (
dx^2)...
         +  (v(i,j+2) - 2*v(i,j+1) + v(i,j))* nu / ( dy^2)...
         - ((0.5.*(v(i,j+1) + v(i,j+2)) ).^2 - (0.5.*(v(i,j) + v(i,j+1)) ).^2)./dy...
         - ((u(i+1,j+1)+u(i+1,j)).*(v(i,j+1)+v(i+1,j+1))...
         - (u(i,j+1)+u(i,j)).*(v(i,j+1)+v(i-1,j+1)))*0.25/dx );
    % Q
    i = 2:Nx-1;
    j = 2:Ny-1;
    Q(i-1,j-1) = 1/dt*((F(i+1,j) - F(i,j))/dx + (G(i,j+1) - G(i,j))/dy);
    % Poisson solve
    p = SOR(Nx,Ny,dx,dy,u,v,p,Q,Re);
%%%%%%%%%% output %%%%%%%%%%
    if (mod(tstep,nout) == 0)
        time = tstep*dt;
        % Quiver Plot
        [uplot, vplot ] = vel_plot(Nx,Ny,u,v);
        figure(1)
        clf,
        quiver(x,y,uplot,vplot,1.2,'k-')
        axis equal; axis([0 Lx 0 Ly])
        title({['2D Cavity Flow with Re = ',num2str(Re)];
            ['time(\itt) = ',num2str(time)]})
        xlabel('Spatial Coordinate (x) \rightarrow')
        ylabel('Spatial Coordinate (y) \rightarrow')
        drawnow;
    end

%%%%%%%%%% update velocity %%%%%%%%%%
%%%%% u velocity
  i = 2:Nx-2;  j = 2:Ny-1;
  u(i+1,j)    = F(i+1,j) - (dt/dx)*(p(i+1,j) - p(i,j));
  u(1,:) = 2*uw - u(2,:); u(end,:)= 2*ue-u(end-1,:);   % Left and rightwall
  u(:,1) = us;            u(:,end) = un;  % Bottom and top wall
%%%%% v velocity
  i = 2:Nx-1;  j = 2:Ny-2;
  v(i,j+1)    = G(i,j+1) - (dt/dy)*(p(i,j+1) - p(i,j));
  v(:,1) = 2*vs - v(:,2); v(:,end)= 2*vn - v(:,end-1);   % Bottom and top wall
  v(1,:) = vw;            v(end,:) = ve;   % Left and right wall
end

%% plot the streamline, pressure contour and velocity field at final time
        figure(2)
        [uplot, vplot ] = vel_plot(Nx,Ny,u,v); skip = 2;
        quiver(x(1:skip:end),y(1:skip:end),uplot(1:skip:end,1:skip:end),...
            vplot(1:skip:end,1:skip:end),1.5,'k-')
        axis equal; axis([0 Lx 0 Ly])
        title({['Velocity field for Re = ',num2str(Re)]})
        xlabel('Spatial Coordinate (x) \rightarrow')
        ylabel('Spatial Coordinate (y) \rightarrow')
        figure(3)
        sx = 0:.02:2;    sy = 0:.02:2;
        fn = stream2(x,y,uplot,vplot,sx,sy);
```

```
        fn1 = stream2(x,y,-uplot,-vplot,sx,sy);
        streamline(fn);  hold on;  streamline(fn1)
        axis equal; axis([0 Lx 0 Ly])
        title({['Streamline for Re = ',num2str(Re)]})
        xlabel('Spatial Coordinate (x) \rightarrow')
        ylabel('Spatial Coordinate (y) \rightarrow')
        figure(4)
        contourf(x,y,p',60,'w-');
        colormap(jet);  colorbar
        axis equal;axis([0 Lx 0 Ly])
        title({['Pressure Contour for Re = ',num2str(Re)]})
        xlabel('Spatial Coordinate (x) \rightarrow')
        ylabel('Spatial Coordinate (y) \rightarrow')
        [yr,ur,xr,vr] = data_literature();    %if one side driven
        %[yr,ur,xr,vr] = data_literature2();  %if two side driven
        %[ur,yr,xr,vr] =analytical_Re1()   %if one side driven Re = 1
         %[ur,yr,xr,vr] =analytical_Re1()  %if one side driven A = 4/3
        figure(5)
        mid = Nx/2+1;
        U(1:Nx,1:Ny)=(1/2)*(u(1:Nx,1:Ny)+ u(2:Nx+1,1:Ny));
        U_center = U(mid,:);plot(U_center,y)
        hold on;            plot(ur,yr,'x')
        legend('program outcome','literature data','location','best');
        title({['Center u for Re = ',num2str(Re)]})
        xlabel('y');ylabel('Centerline u') ; hold off
        figure(6)
        mid = Ny/2+1;
        V(1:Nx,1:Ny)=(1/2)*(v(1:Nx,1:Ny) + v(1:Nx,2:Ny+1));
        V_center = V(:,mid); plot(x,V_center)
        hold on;             plot(xr,vr,'x')
        legend('program outcome','literature data','location','best');
        title({['Center v for Re = ',num2str(Re)]})
        xlabel('x');ylabel('Centerline v') ; hold off
%  can use xlswrite to save u,v,p results for convergence study %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [uplot,vplot] = vel_plot(Nx,Ny,u,v)
   uplot(1:Nx,1:Ny)  = (1/2) * (u(1:Nx,1:Ny) + u(2:Nx+1,1:Ny));
   vplot(1:Nx,1:Ny)  = (1/2) * (v(1:Nx,1:Ny) + v(1:Nx,2:Ny+1));
   Len   = sqrt(uplot.^2 + vplot.^2 + eps);
   uplot = (uplot./Len)';   vplot = (vplot./Len)';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     poisson solve    %
function Successive_Over_Relaxation = SOR(Nx,Ny,dx,dy,u,v,p,Q,Re)
    % Poisson matrix
    rf    = 1.6;      epsi   = 0.001;      itmax = 5000;
    change = 2*epsi;
    it     = 1;
    while (change > epsi)
      pold = p;
    % boundary condition
        p(2:end-1 ,1    ) = p(2:end-1 ,2  ) - 2.0*v(2:end-1,3 )/(Re*dy); % bottom
        p(2:end-1 ,end  ) = p(2:end-1 ,end-1 ) + 2.0*v(2:end-1 ,end-2 )/(Re*dy); % top
        p(1   ,2:end-1) = p(2   ,2:end-1) - 2.0*u(3 ,2:end-1 )/(Re*dx); % left
        p(end ,2:end-1) = p(end-1 ,2:end-1) + 2.0*u(end-2 ,2:end-1 )/(Re*dx); % right
      % SOR
        for i=2:Nx-1
            for j=2:Ny-1
                p(i,j) = 0.25*(p(i-1,j)+p(i,j-1)+p(i+1,j)+p(i,j+1) - ...
                            Q(i-1,j-1)*dx^2);
                p(i,j) = pold(i,j) + rf*(p(i,j)-pold(i,j));
            end
        end
        pmax = max(abs(pold(:)));
```

```
        if (pmax == 0) , pmax = 1.0;  end
        change =  max(abs( (pold(:)- p(:))/pmax ));
        it = it + 1;
        if (it > itmax) , break;  end
        Successive_Over_Relaxation = p;
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [x1,y1,x4,y4] = data_literature()     % center velocity for one side driven
load data1.dat;
load data4.dat;
x1 = data1(:,2);y1 = data1(:,1);
x4 = data4(:,1);y4 = data4(:,2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [x1,y1,x4,y4] = data_literature2()     % center velocity for two side driven
load data2.dat;
load data3.dat;
x1 = data2(:,2);y1 = data2(:,1);
x4 = data3(:,1);y4 = data3(:,2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [x1,y1,x4,y4] = analytical_Re1()     % center velocity for one side driven Re
= 1
h = 1/32;
y1 = 0:h:1;  x1 = (4*y1.^3-2*y1).*0.5;
x4 = 0:h:1;  y4 = (4*x4.^3-6*x4.^2+2*x4).*1.5;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [x1,y1,x4,y4] = data_literature3()     % center velocity for one side driven
% aspect ratio A = 4/3 %
load data5.dat;
load data6.dat;
x1 = data5(:,1);y1 = data5(:,2);
x4 = data6(:,1);y4 = data6(:,2);
```

Convergence Study

```
%% Time discretization error

function dt_convergence()

clc; clear all;  close all;  %final time = 0.1
timestep = [1e-2 1e-3 1e-4 1e-5 1e-6];   ep = [1:5]*0;
p1 = xlsread('p for 128by128 and dt = 0.01.xlsx');
p2 = xlsread('p for 128by128 and dt = 0.01e-1.xlsx');
p3 = xlsread('p for 128by128 and dt = 0.01e-2.xlsx');
p4 = xlsread('p for 128by128 and dt = 0.01e-3.xlsx');
p5 = xlsread('p for 128by128 and dt = 0.01e-4.xlsx');
pr = xlsread('p for 128by128 and dt = 0.01e-5.xlsx');
a = reshape(p1,1,[]);b=reshape(pr,1,[]);ep(1) = rmse(a,b);
a = reshape(p2,1,[]);b=reshape(pr,1,[]);ep(2) = rmse(a,b);
a = reshape(p3,1,[]);b=reshape(pr,1,[]);ep(3) = rmse(a,b);
a = reshape(p4,1,[]);b=reshape(pr,1,[]);ep(4) = rmse(a,b);
a = reshape(p5,1,[]);b=reshape(pr,1,[]);ep(5) = rmse(a,b);
figure()
plot(timestep,ep); title('timestep vs p error');xlabel('dt'); ylabel('p error')
figure()
loglog(timestep,ep); title('loglog timestep vs p error')
xlabel('log dt'); ylabel('log p error')
grid on; grid minor

a = reshape(p1,1,[]);b=reshape(pr,1,[]);ep(1) = rmse(a,b);
```

```
u1 = xlsread('u for 128by128 and dt = 0.01.xlsx');
u2 = xlsread('u for 128by128 and dt = 0.01e-1.xlsx');
u3 = xlsread('u for 128by128 and dt = 0.01e-2.xlsx');
u4 = xlsread('u for 128by128 and dt = 0.01e-3.xlsx');
u5 = xlsread('u for 128by128 and dt = 0.01e-4.xlsx');
ur = xlsread('u for 128by128 and dt = 0.01e-5.xlsx');
eu = [1:5]*0;
a = reshape(u1,1,[]);b=reshape(ur,1,[]);ep(1) = rmse(a,b);
a = reshape(u2,1,[]);b=reshape(ur,1,[]);ep(2) = rmse(a,b);
a = reshape(u3,1,[]);b=reshape(ur,1,[]);ep(3) = rmse(a,b);
a = reshape(u4,1,[]);b=reshape(ur,1,[]);ep(4) = rmse(a,b);
a = reshape(u5,1,[]);b=reshape(ur,1,[]);ep(5) = rmse(a,b);
figure()
plot(timestep,eu);title('timestep vs u error')
figure()
loglog(timestep,eu);title('loglog timestep vs u error')
xlabel('log dt'); ylabel('log u error')
grid on; grid minor

v1 = xlsread('v for 128by128 and dt = 0.01.xlsx');
v2 = xlsread('v for 128by128 and dt = 0.01e-1.xlsx');
v3 = xlsread('v for 128by128 and dt = 0.01e-2.xlsx');
v4 = xlsread('v for 128by128 and dt = 0.01e-3.xlsx');
v5 = xlsread('v for 128by128 and dt = 0.01e-4.xlsx');
vr = xlsread('v for 128by128 and dt = 0.01e-5.xlsx');
ev = [1:5]*0;
a = reshape(v1,1,[]);b=reshape(vr,1,[]);ep(1) = rmse(a,b);
a = reshape(v2,1,[]);b=reshape(vr,1,[]);ep(2) = rmse(a,b);
a = reshape(v3,1,[]);b=reshape(vr,1,[]);ep(3) = rmse(a,b);
a = reshape(v4,1,[]);b=reshape(vr,1,[]);ep(4) = rmse(a,b);
a = reshape(v5,1,[]);b=reshape(vr,1,[]);ep(5) = rmse(a,b);
figure()
plot(timestep,ev);title('timestep vs u error')
figure()
loglog(timestep,ev);title('loglog timestep vs v error')
xlabel('log dt'); ylabel('log v error')
grid on; grid minor
%%Spatial discretization error
function dx_convergence()
clc; clear all;  close all;    %%dt = 5e-5 and final time = 0.1
dx = [1/32 1/64 1/128 1/256 1/512];
p1 = xlsread('p for 32by32.xlsx');
p2 = xlsread('p for 64by64.xlsx');
p3 = xlsread('p for 128by128.xlsx');
p4 = xlsread('p for 256by256.xlsx');
p5 = xlsread('p for 512by512.xlsx');
pr = xlsread('p for 1024by1024.xlsx');

u1 = xlsread('u for 32by32.xlsx');
u2 = xlsread('u for 64by64.xlsx');
u3 = xlsread('u for 128by128.xlsx');
u4 = xlsread('u for 256by256.xlsx');
u5 = xlsread('u for 512by512.xlsx');
ur = xlsread('u for 1024by1024.xlsx');

v1 = xlsread('v for 32by32.xlsx');
v2 = xlsread('v for 64by64.xlsx');
v3 = xlsread('v for 128by128.xlsx');
v4 = xlsread('v for 256by256.xlsx');
v5 = xlsread('v for 512by512.xlsx');
vr = xlsread('v for 1024by1024.xlsx');
```

```
ep = [1:5]*0;
a = reshape(p1(2:end-1,2:end-1),1,[]);b=reshape(pr(2:32:end-1,2:32:end-1),1,[]);ep(1) =
rmse(a,b);
a = reshape(p2(2:end-1,2:end-1),1,[]);b=reshape(pr(2:16:end-1,2:16:end-1),1,[]);ep(2) =
rmse(a,b);
a = reshape(p3(2:end-1,2:end-1),1,[]);b=reshape(pr(2:8:end-1,2:8:end-1),1,[]);ep(3) = rmse(a,b);
a = reshape(p4(2:end-1,2:end-1),1,[]);b=reshape(pr(2:4:end-1,2:4:end-1),1,[]);ep(4) = rmse(a,b);
a = reshape(p5(2:end-1,2:end-1),1,[]);b=reshape(pr(2:2:end-1,2:2:end-1),1,[]);ep(5) = rmse(a,b);
figure()
plot(dx,ep);title('timestep vs p error')
figure()
loglog(dx,ep);title('loglog dx vs p error')
xlabel('log dx'); ylabel('log p error')
grid on; grid minor

eu = [1:5]*0;
a = reshape(u1(2:end-2,2:end-1),1,[]);b=reshape(ur(2:32:end-2,2:32:end-1),1,[]);eu(1) =
rmse(a,b);
a = reshape(u2(2:end-2,2:end-1),1,[]);b=reshape(ur(2:16:end-2,2:16:end-1),1,[]);eu(2) =
rmse(a,b);
a = reshape(u3(2:end-2,2:end-1),1,[]);b=reshape(ur(2:8:end-2,2:8:end-1),1,[]);eu(3) = rmse(a,b);
a = reshape(u4(2:end-2,2:end-1),1,[]);b=reshape(ur(2:4:end-2,2:4:end-1),1,[]);eu(4) = rmse(a,b);
a = reshape(u5(2:end-2,2:end-1),1,[]);b=reshape(ur(2:2:end-2,2:2:end-1),1,[]);eu(5) = rmse(a,b);
figure()
plot(dx,eu);title('timestep vs u error')
figure()
loglog(dx,eu);title('loglog dx vs u error')
xlabel('log dx'); ylabel('log u error')
grid on; grid minor

ev = [1:5]*0;
a = reshape(v1(2:end-1,2:end-2),1,[]);b=reshape(vr(2:32:end-1,2:32:end-2),1,[]);ev(1) =
rmse(a,b);
a = reshape(v2(2:end-1,2:end-2),1,[]);b=reshape(vr(2:16:end-1,2:16:end-2),1,[]);ev(2) =
rmse(a,b);
a = reshape(v3(2:end-1,2:end-2),1,[]);b=reshape(vr(2:8:end-1,2:8:end-2),1,[]);ev(3) = rmse(a,b);
a = reshape(v4(2:end-1,2:end-2),1,[]);b=reshape(vr(2:4:end-1,2:4:end-2),1,[]);ev(4) = rmse(a,b);
a = reshape(v5(2:end-1,2:end-2),1,[]);b=reshape(vr(2:2:end-1,2:2:end-2),1,[]);ev(5) = rmse(a,b);
figure()
plot(dx,ev);title('timestep vs v error')
figure()
loglog(dx,ev);title('loglog dx vs v error')
xlabel('log dx'); ylabel('log v error')
grid on; grid minor

% for point(0.5, 0.95)
ep2 = [1:5]*0;          b = pr(ceil(end*0.5),ceil(end*0.95));
a = p1(ceil(end*0.5),ceil(end*0.95));ep2(1)=abs(a-b);
a = p2(ceil(end*0.5),ceil(end*0.95));ep2(2)=abs(a-b);
a = p3(ceil(end*0.5),ceil(end*0.95));ep2(3)=abs(a-b);
a = p4(ceil(end*0.5),ceil(end*0.95));ep2(4)=abs(a-b);
a = p5(ceil(end*0.5),ceil(end*0.95));ep2(5)=abs(a-b);
figure()
plot(dx,ep2);title('timestep vs p error')
figure()
loglog(dx,ep2);title('loglog dx vs p error')
xlabel('log dx'); ylabel('log p error');grid on; grid minor

eu2 = [1:5]*0;          b = ur(ceil(end*0.5),ceil(end*0.95));
a = u1(ceil(end*0.5),ceil(end*0.95));eu2(1)=abs(a-b);
a = u2(ceil(end*0.5),ceil(end*0.95));eu2(2)=abs(a-b);
a = u3(ceil(end*0.5),ceil(end*0.95));eu2(3)=abs(a-b);
a = u4(ceil(end*0.5),ceil(end*0.95));eu2(4)=abs(a-b);
a = u5(ceil(end*0.5),ceil(end*0.95));eu2(5)=abs(a-b);
figure()
```

```
plot(dx,eu2);title('timestep vs u error')
figure()
loglog(dx,eu2);title('loglog dx vs u error')
xlabel('log dx'); ylabel('log u error');grid on; grid minor

ev2 = [1:5]*0;          b = vr(ceil(end*0.5),ceil(end*0.95));
a = v1(ceil(end*0.5),ceil(end*0.95));ev2(1)=abs(a-b);
a = v2(ceil(end*0.5),ceil(end*0.95));ev2(2)=abs(a-b);
a = v3(ceil(end*0.5),ceil(end*0.95));ev2(3)=abs(a-b);
a = v4(ceil(end*0.5),ceil(end*0.95));ev2(4)=abs(a-b);
a = v5(ceil(end*0.5),ceil(end*0.95));ev2(5)=abs(a-b);
figure()
plot(dx,ev2);title('timestep vs v error')
figure()
loglog(dx,ev2);title('loglog dx vs v error')
xlabel('log dx'); ylabel('log v error');grid on; grid minor


function r=rmse(data,estimate)

 I = ~isnan(data) & ~isnan(estimate);
 data = data(I); estimate = estimate(I);
 r=sqrt(sum(((data(:)-estimate(:)).^2)/numel(data)));
```

Analytical solution for the main vortex at Re = 1

```
function analytical_solution()
 N = 1024
 X = ( linspace ( 0, 1, N ) );
 Y = ( linspace ( 0, 1, N) );
 [x,y] = meshgrid(X, Y);
 u = (x.^4-2*x.^3+x.^2).*(4*y.^3-2*y).*8;
 v = -(4*x.^3-6*x.^2+2*x).*(y.^4-y.^2).*8;
  skip = 32;

 figure()
 quiver(x(1:skip:N,1:skip:N),y(1:skip:N,1:skip:N),u(1:skip:N,1:skip:N),v(1:skip:N,1:skip:N),2)
 xlabel('x'); ylabel('y'); title('velocity field for Re = 1'); axis equal; axis([0 1 0 1])
```

Reference

[1] AE 601 lecture notes from Dr. Gustaaf Jacobs

[2] Ghia, U.K.N.G., Ghia, K.N. and Shin, C.T., 1982. High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *Journal of computational physics*, *48*(3), pp.387-411.

[3] Marchi, C.H., Suero, R. and Araki, L.K., 2009. The lid-driven square cavity flow: numerical solution with a 1024 x 1024 grid. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, *31*(3), pp.186-198.

[4] Perumal, A. and Dass, A.K., 2010. Simulation of Incompressible Flows in Two-Sided Lid-Driven Square Cavities. Part I-FDM. *CFD Letters*, *2*(1), pp.13-24.

[5] Shih, T.M., Tan, C.H. and Hwang, B.C., 1989. Effects of grid staggering on numerical schemes. *International Journal for numerical methods in fluids*, *9*(2), pp.193-212.

[6] Harlow, F.H. and Welch, J.E., 1965. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of fluids*, *8*(12), p.2182.