MAE 6263 Course Project 3

S M Abdullah Al Mamun
CWID : A20138451

May 13, 2018

1 Introduction

The purpose of this course project is to document the development of numerical routine and its solution for the 2-D Navier-Stokes equations for a configuration of 2-D channel flow. An explicit iterative scheme shall be devised with two different methods used to treat the pressure Poisson's equation. Through this process it is endeavored to understand the importance of the use of a pressure solver that ensures a divergence free field throughout the solution process. A simple compact stencil for the pressure is used to demonstrate how the divergence increases throughout the system but momentum interpolation method ensure a divergence free field.

1.1 Problem Statement

The 2-D Navier-Stokes equations can be written as follows

$$\frac{\partial u}{\partial t} + \bar{u}.\nabla\bar{u} = \nu\Delta\bar{u} - \frac{1}{\rho}\nabla p \qquad (1.1)$$

$$\nabla.\bar{u} = 0 \qquad (1.2)$$

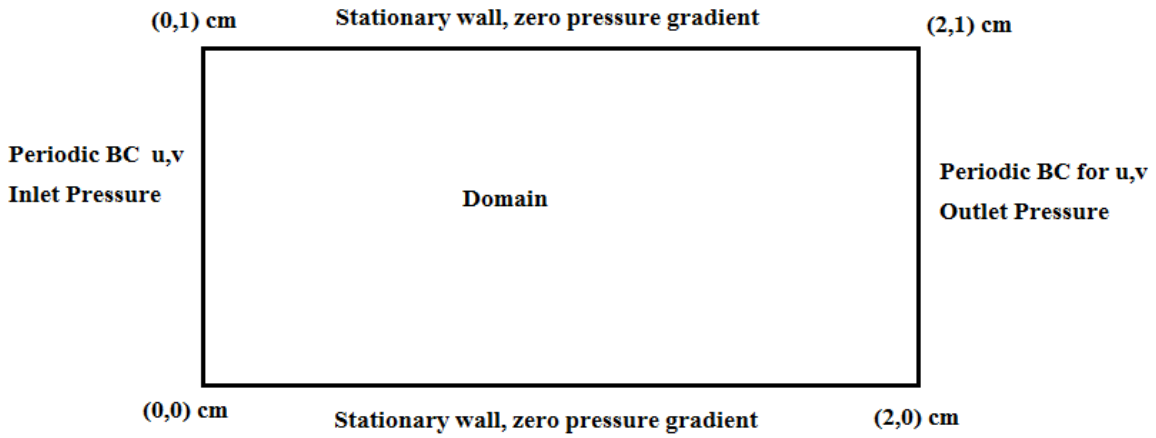
The evolution equation for pressure to complete the set of equations required to solve for the fluid flow solution in a given domain is obtained by the divergence of this equation and is detailed later on in the document.

The domain to be solved is rectangular domain of unit width and length two units which have initial and boundary conditions in velocity specified by:

$$At\ t = 0 \left\{ \begin{matrix} \bar{u}\,(x,y) = 0\ for\ 0 \le x \le 2, 0 \le y \le 1; \\ p(x,y) = 0\ everywhere\ except\ boundaries \end{matrix} \right\}$$

$$BC: \begin{Bmatrix} \bar{u}(x,y) = 0 \; for \; y = 1; \\ \bar{u}(x,y) = 0 \; for \; y = 0; \\ \bar{u}(x,y) = \bar{u}(x-2,y) \; for \; x = 2; \\ \bar{u}(x,y) = \bar{u}(x+2,y) \; for \; x = 0; \end{Bmatrix}$$

$$BC: \begin{Bmatrix} p(x,y) = p_{outlet} \; for \; x = 2; \\ p(x,y) = p_{inlet} \; for \; x = 0; \\ \frac{\partial p}{\partial y} = 0 \; for \; y = 0 \; and \; y = 1 \end{Bmatrix}$$



Fig(1.1) : Considered Domain for 2D flow

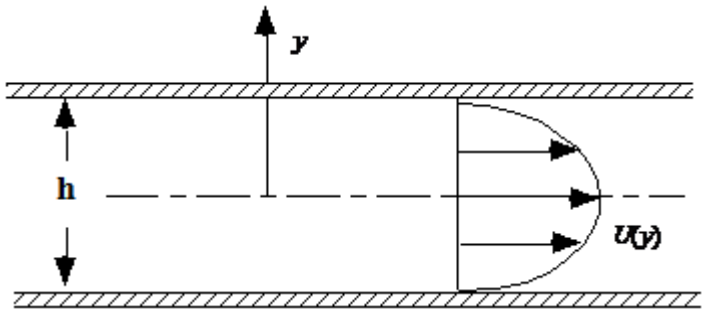
The exact solution of this system is given by the Poiseuille equation which prescribes a parabolic flow for the x component of the velocity on variation of the wall normal displacement. The exact solution of this domain and boundary condition is given by –

$$u(y) = -\frac{1}{2\mu}\frac{\partial p}{\partial x} y(h-y) \qquad (1.3)$$

Where h is the width of the channel and y is the distance of the point in question from the center-line. Using this expression and given values in the problem we get $\partial p = -0.16$

Therefore inlet pressure can be expressed as –

$$p_{inlet} = p_{outlet} - (-0.16)$$



Fig(1.2) : Expected steady state profile for velocity

1.2 Derivation of Pressure Poisson Equation

Pressure has no equation of its own in the original system of equation. However, the continuity and momentum equations can be manipulated to yield a Poisson equation governing the pressure field. The pressure Poisson equation can be derived by taking divergence of the momentum equation.

The momentum equation can be written in semi-discrete formulation as

$$\frac{\mathbf{u}^{n+1}-\mathbf{u}^{n}}{\Delta t} = -\mathbf{u}^{n}\cdot\nabla\mathbf{u}^{n} - \nabla p^{n} + \frac{1}{\mathrm{Re}}\nabla^{2}\mathbf{u}^{n}$$

Now the velocity at the new time level can be obtained as

$$\mathbf{u}^{n+1} = \mathbf{u}^{n} + \Delta t\left\{-\mathbf{u}^{n}\cdot\nabla\mathbf{u}^{n} - \nabla p^{n} + \frac{1}{\mathrm{Re}}\nabla^{2}\mathbf{u}^{n}\right\}$$

Where pressure, p is not known. Since $\mathbf{u}^{n+1}$ should satisfy continuity, we apply the gradient operator $(\nabla\cdot)$ to the above equation to write

$$\nabla\cdot\left[\mathbf{u}^{n+1} = \mathbf{u}^{n} + \Delta t\left\{-\mathbf{u}^{n}\cdot\nabla\mathbf{u}^{n} - \nabla p^{n} + \frac{1}{\mathrm{Re}}\nabla^{2}\mathbf{u}^{n}\right\}\right] = 0$$

Assuming that the continuity equation was enforced at the old time level, i.e $\nabla\mathbf{u}^{n} = 0$, we obtain

$$\nabla^2 p^n = \nabla \cdot \left[-\mathbf{u}^n \cdot \nabla \mathbf{u}^n + \frac{1}{\mathrm{Re}} \nabla^2 \mathbf{u}^n \right]$$

The above equation is known as pressure Poisson equation.

If we consider $\mathbf{H} = -\mathbf{u}^n \cdot \nabla \mathbf{u}^n + \frac{1}{\mathrm{Re}} \nabla^2 \mathbf{u}^n$ then the pressure Poisson equation takes the form $\nabla^2 p^n = \nabla \cdot \mathbf{H}$

1.3 Explicit Solution Algorithm

Using the semi-discrete version of the equations we discussed in previous section the overall solution procedure for the explicit scheme can be given as follows

1. Start with velocity field $u^n @ t_n$ (assumed divergence free)
2. Compute $\mathbf{H}^n$ ($\mathbf{H}^n = -\mathbf{u}^n \cdot \nabla \mathbf{u}^n + \frac{1}{\mathrm{Re}} \nabla^2 \mathbf{u}^n$)
3. Compute $\frac{\partial p^n}{\partial N}\Big|_{boundary}$ (for Cartesian grid this will be $\frac{\partial P}{\partial x}$, $\frac{\partial p}{\partial y}$
4. Solve pressure Poisson equation to get p^n
5. Compute $\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t(\mathbf{H}^n - -\nabla p^n)$
6. Continue this with time step until steady state is reached.

1.4 Conservation of Kinetic Energy

The numerical schemes used for the gradient and divergence scheme are a central difference scheme for both. In order to prove kinetic energy conservation, let us assume a hypothetical 1D domain with nodes numbered from 0 to 5 (i.e. nodes 0 and 5 are at the surface of the domain). Kinetic energy conservation can be confirmed if the following expression holds true.

$$\sum_{i=1}^{4} (u_i G_i p + p_i D_i u_i) \Delta \Omega = \text{surface terms only} \qquad (1.4)$$

Assuming a central difference scheme for both gradient and divergence operations we can obtain the following extended expression for the assumed 1D domain –

$$u_1(p_2 - p_0) + p_1(u_2 - u_0) + u_2(p_3 - p_1) + p_2(u_3 - u_1) + u_3(p_4 - p_2) + p_3(u_4 - u_2) + u_4(p_5 - p_3) + p_4(u_5 - u_3) = -u_1p_0 - p_1u_0 + p_4u_5 + u_4p_5$$

From the above equation it can be seen that only surface terms are left behind and so the choice of gradient and divergence schemes for this project will conserve kinetic energy.

1.5 Compact Stencil Pressure Solver

A compact stencil may be used to solve the pressure Poisson equation as given in the handouts. The explicit scheme considering Forward Euler for time derivative, central difference for the viscous term and the first order upwind for the advective term to solve the momentum equation for the velocity is shown below-

$$\frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\Delta t} = -u_{i,j}\frac{u_{i,j}^{n} - u_{i-1,j}^{n}}{\Delta x} - v_{i,j}\frac{u_{i,j}^{n} - u_{i,j-1}^{n}}{\Delta y} - \frac{p_{i+1,j}^{n} - p_{i-1,j}^{n}}{2\Delta x} + \frac{1}{Re}\frac{u_{i+1,j}^{n} + u_{i-1,j}^{n} - 2u_{i,j}^{n}}{\Delta x^2} + \frac{1}{Re}\frac{u_{i,j+1}^{n} + u_{i,j-1}^{n} - 2u_{i,j}^{n}}{\Delta y^2} \quad (1.5)$$

Similarly we have

$$\frac{v_{i,j}^{n+1} - v_{i,j}^{n}}{\Delta t} = -u_{i,j}\frac{v_{i,j}^{n} - v_{i-1,j}^{n}}{\Delta x} - v_{i,j}\frac{v_{i,j}^{n} - v_{i,j-1}^{n}}{\Delta y} - \frac{p_{i,j+1}^{n} - p_{i,j-1}^{n}}{2\Delta y} + \frac{1}{Re}\frac{v_{i+1,j}^{n} + v_{i-1,j}^{n} - 2v_{i,j}^{n}}{\Delta x^2} + \frac{1}{Re}\frac{v_{i,j+1}^{n} + v_{i,j-1}^{n} - 2v_{i,j}^{n}}{\Delta y^2} \quad (1.6)$$

The compact stencil implementation of the pressure Poisson equation is given by –

$$\frac{p_{i+1,j}^{n} - 2p_{i,j}^{n} + p_{i-1,j}^{n}}{\Delta x^2} + \frac{p_{i,j+1}^{n} - 2p_{i,j}^{n} + p_{i,j-1}^{n}}{\Delta y^2} = \frac{Hx_{i+1,j}^{n} - Hx_{i-1,j}^{n}}{2\Delta x} + \frac{Hy_{i,j+1}^{n} - Hy_{i,j-1}^{n}}{2\Delta y} \quad (1.7)$$

The above equation simplifies considerably due to the selection of an equal grid spacing in both *x* and *y* directions. It should be noted that the above method is not at all ideal to solve the NS equations with due to the fact that there is no mechanism to ensure the successively evolved fields through the explicit algebraic scheme are divergence free.

This would inevitably lead to an increase (or decrease) in the mass of the system eventually leading to a physically incorrect solution.

1.5 Momentum Interpolation Method

For p control volume, we need to compute face velocity. One of the approaches for estimating the cell face velocities is the so-called momentum interpolation method. It involves the estimation of the interface velocity by writing an interfacial u-momentum equation similar to the nodal u-momentum equation.

The equation for the nodal velocity can be written like below

$$u_p = H_p - \frac{\Delta\Omega_p}{A_p^p}\left(\frac{\partial p}{\partial x}\right)_p \tag{1.8}$$

Now we can write a similar momentum equation for u_e for east face

$$u_e = H_e - \frac{\Delta\Omega_e}{A_p^e}\left(\frac{\partial p}{\partial x}\right)_e \tag{1.9}$$

Where H_e is estimated by interpolation of the nodal values:

$$H_e = \frac{1}{2}\left(H_p + H_E\right) \tag{1.10}$$

The crucial idea in the momentum interpolation method is that the pressure gradient term can be written as

$$\left(\frac{\partial p}{\partial x}\right)_e = \frac{P_E - P_E}{\Delta x} \tag{1.11}$$

Thus the interface velocity takes the form

$$u_e = \frac{1}{2}(u_P + u_E) + \frac{1}{2}\left[\left(\frac{\Delta\Omega_P}{A_p^P} + \frac{\Delta\Omega_E}{A_p^E}\right)(P_P - P_E) - \frac{\Delta\Omega_P}{A_p^P}(P_E - P_W) - \frac{\Delta\Omega_E}{A_p^E}(P_{EE} - P_P)\right] \tag{1.12}$$

The pressure Poisson equation is obtained by substituting the interface velocity into the continuity equation. The pressure smoothing term in the above equation helps to maintain mass conservation. This method also results in a compact stencil and thus provides strong coupling between pressure-velocity and avoids odd-even oscillations.

2. Numerical Experimentation

Our pressure Poisson equation is

$$\frac{p_{i+1,j}^{n} - 2p_{i,j}^{n} + p_{i-1,j}^{n}}{\Delta x^2} + \frac{p_{i,j+1}^{n} - 2p_{i,j}^{n} + p_{i,j-1}^{n}}{\Delta y^2} = \frac{Hx_{i+1,j}^{n} - Hx_{i-1,j}^{n}}{2\Delta x} + \frac{Hy_{i,j+1}^{n} - Hy_{i,j-1}^{n}}{2\Delta y}$$

We can consider the right side part as a constant for the loop of pressure Poisson solver.

Let $H_c = \frac{Hx_{i+1,j}^{n} - Hx_{i-1,j}^{n}}{2\Delta x} + \frac{Hy_{i,j+1}^{n} - Hy_{i,j-1}^{n}}{2\Delta y}$

Then we can form matrix **A** and vector **B** considering it as a linear system to solve.

Where **A** =

$$\begin{bmatrix}
-2\left(\frac{1}{\Delta x\text{^}2} + \frac{1}{\Delta y\text{^}2}\right) & \frac{1}{\Delta x\text{^}2} & 0\cdots & \frac{1}{\Delta y\text{^}2} & \cdots & 0 \\
\frac{1}{\Delta x\text{^}2} & -2\left(\frac{1}{\Delta x\text{^}2} + \frac{1}{\Delta y\text{^}2}\right) & \frac{1}{\Delta x\text{^}2} & 0 & \cdots & \frac{1}{\Delta y\text{^}2} \\
0 & \frac{1}{\Delta x\text{^}2} & -2\left(\frac{1}{\Delta x\text{^}2} + \frac{1}{\Delta y\text{^}2}\right) & \ddots & \cdots & \vdots \\
\vdots & 0 & \ddots & \ddots & \ddots & 0 \\
\frac{1}{\Delta y\text{^}2} & \vdots & \vdots & \ddots & \ddots & \frac{1}{\Delta x\text{^}2} \\
0 & \frac{1}{\Delta y\text{^}2} & \cdots & 0 & \frac{1}{\Delta x\text{^}2} & -2\left(\frac{1}{\Delta x\text{^}2} + \frac{1}{\Delta y\text{^}2}\right)
\end{bmatrix}$$

Where **B** =

$$\begin{bmatrix} P_{11} \\ P_{21} \\ P_{31} \\ \vdots \\ P_{n_x 1} \\ P_{12} \\ P_{22} \\ P_{32} \\ \vdots \\ P_{n_x 2} \\ \vdots \\ P_{n_x n_y} \end{bmatrix}$$

I couldn't be able to make the code of momentum interpolation method work so far that's why I'm presenting the results and analysis for the without momentum interpolation method here only.

First we will present the steady state iso-contours of the pressure and velocity field.

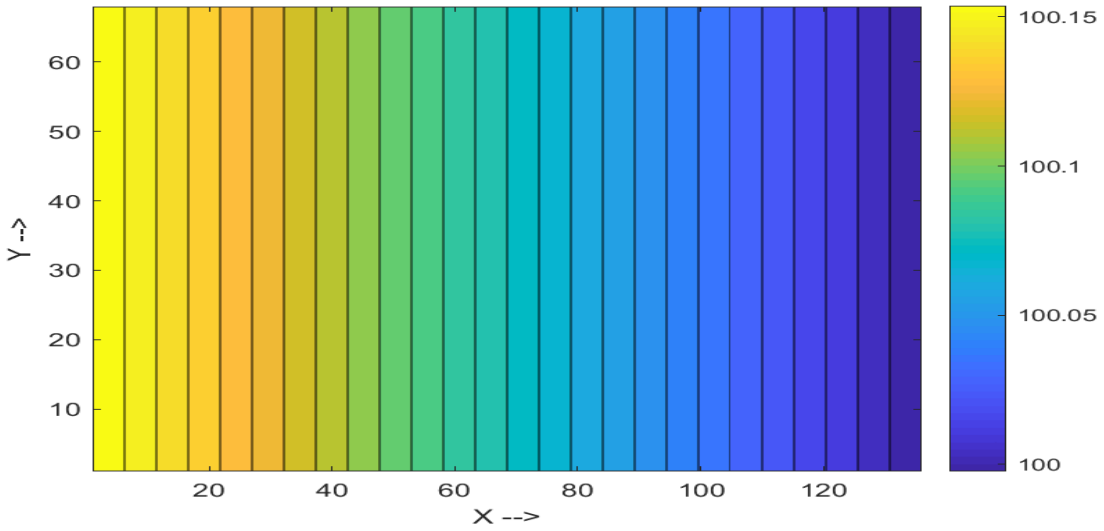


Fig (2.1) : Steady state iso-contours for pressure field (without momentum interpolation)

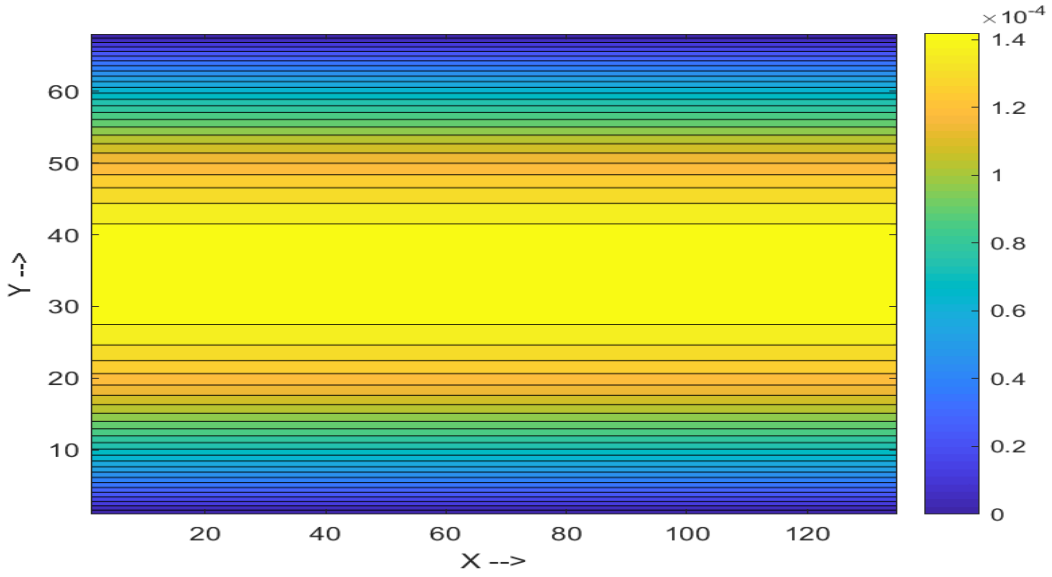


Fig (2.2) : Steady state iso-contours for x component of velocity field (without momentum interpolation)

As we can see it is difficult to determine any deficit from the steady state contour. It can be interpreted in better way by divergence tracking.

The following plots represents x component of velocity at x=1.0 vs y progressing from initial state to steady state and the plot for pressure too.

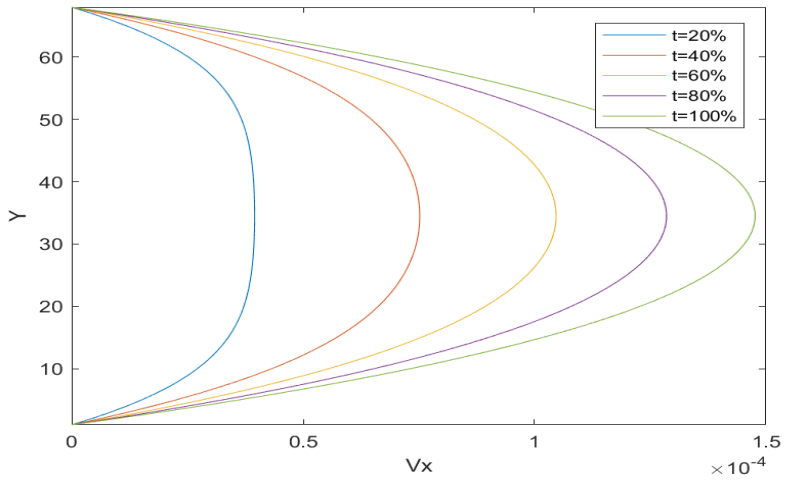


Fig (2.3) : Line plot for x component of velocity at x=1.0 (without momentum interpolation)

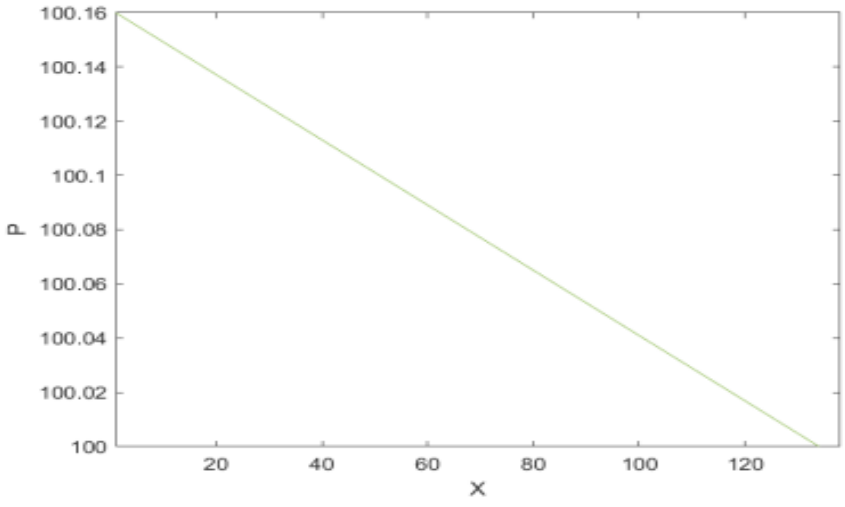


Fig (2.4) : Line plot for pressure (without momentum interpolation)

Here it is observed that it validates the parabolic velocity profile at steady state.

The mass in the system can be tracked by plotting the divergence with iteration number on progress to the steady state.

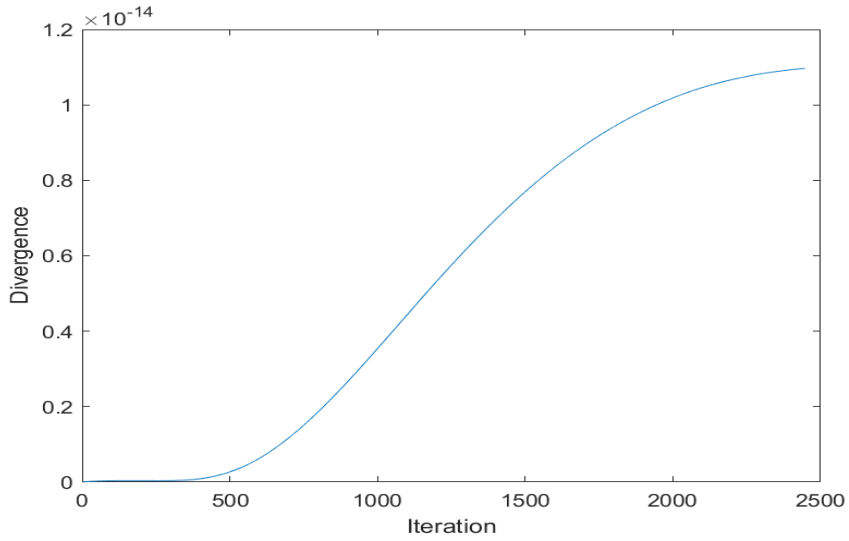


Fig (2.5) : Divergence vs Iteration plot (without momentum interpolation)

Here we can see that in compact scheme without momentum interpolation a non-zero divergence evolves.

The energy in the system is plotted below as a function of iteration number as the code converges to steady state. The energy of the system is plotted by the following summation –

$$\sum_{i=1}^{n_x}\sum_{j=1}^{n_y}\rho u_{i,j}^2$$

Where n_x and n_y are the total number points in the x and y directions respectively.

The plot for the Energy is given below –

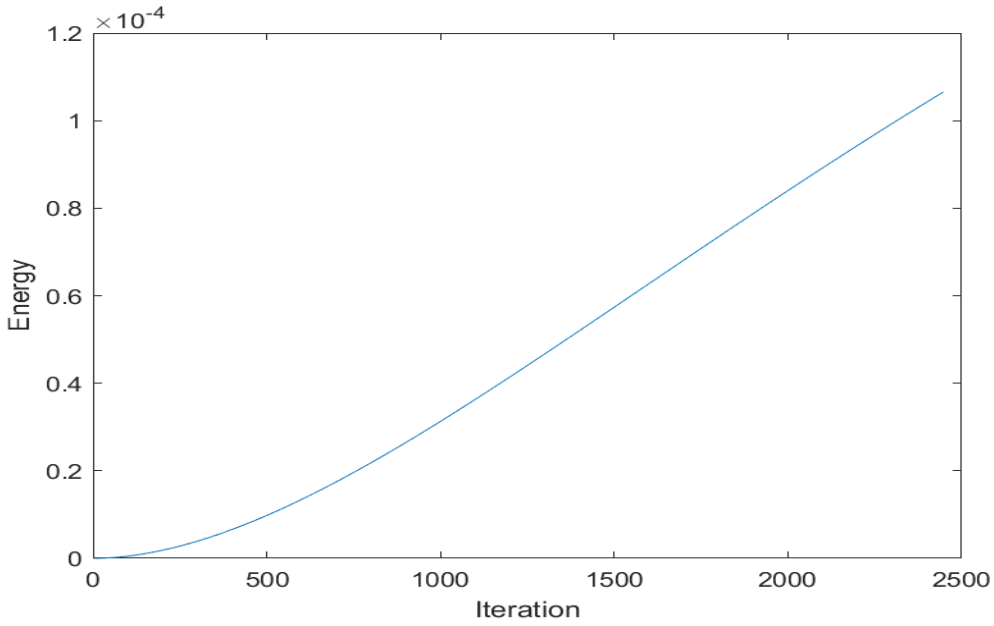


Fig (2.6) : Energy vs Iteration plot (without momentum interpolation)

From here we can observe that energy in the system is going to a fixed value as the system attain steady state.

```
%%%%%%% 2D NS equation Solver %%%%%%

Re=100;
Ly=1;
Lx=2;
neu=1e-2;
ro=1;
Vpeak = (Re*neu)/Ly ;
Pout = 100;
Q = Vpeak*Ly;
meu = neu*ro;
dp = -0.16 ;
Pin = Pout - dp ;

MI = 0;            % MI = 1 for with momentum
interpolation and 0 for without

gamma = 0.2;
Pe = 1.5;
delx = (Pe*neu)/Vpeak;
dely = delx ;
delt = (gamma*delx^2)/neu ;

nx = round(Lx/delx) + 1 ;
ny = round(Ly/dely) + 1 ;

u = zeros(nx,ny);
v = zeros(nx,ny);
P = zeros(nx,ny);
Hx = zeros(nx,ny);
Hy = zeros(nx,ny);
E = zeros (2447,1);
div = zeros (2447,1);

for i = 1:nx
    for j = 1:ny
        P(1,j)= Pin ;
        P(nx,j)= Pout;
        end
end
```

```
for i = 2:nx-1
    for j = 1:ny
P(i,j) = Pin + dp*(i-1)/(nx-1);
    end
end
P(nx+1,:)=Pout + dp/(nx-1);
P(nx+2,:)=Pout + (2*dp/(nx-1));

P1 = zeros(ny,1);
P1(:,1)= Pin - dp/(nx-1);


norm_u=100;
norm_v=100;
tol_vel = 1e-4;
time_iter = 0;

while (norm_u > tol_vel || norm_v > tol_vel)
time_iter = time_iter+1;

u(1,:) = u(nx,:);
u(nx+1,:)=u(2,:);
v(1,:) = v(nx,:);
v(nx+1,:)=v(2,:);
Hx(1,:) = Hx(nx,:);
Hx(nx+1,:)=Hx(2,:);
Hy(1,:) = Hy(nx,:);
Hy(nx+1,:)=Hy(2,:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i = 2:nx
    for j = 2:ny-1

Hx(i,j) = -(u(i,j)*((u(i,j)-u(i-
1,j))/delx)+v(i,j)*((u(i,j)-u(i,j-1))/dely))...
    +(1/Re)*((u(i+1,j)-2*u(i,j)+u(i-1,j))/delx^2 ...
    +(u(i,j+1)-2*u(i,j)+u(i,j-1))/dely^2) ;

Hy(i,j) = -(u(i,j)*((v(i,j)-v(i-
1,j))/delx)+v(i,j)*((v(i,j)-v(i,j-1))/dely))...
    +(1/Re)*((v(i+1,j)-2*v(i,j)+v(i-1,j))/delx^2 ...
    +(v(i,j+1)-2*v(i,j)+v(i,j-1))/dely^2) ;
```

```
    end
end
err = 0;
norm = 100;
tol = 1e-5;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%% Without Momentum %%%%%%%%
if (MI == 0)

while (norm > tol)

Pold = P;
for i = 2:nx-1
    for j = 2:ny-1

P(i,j)=((Hx(i+1,j)-Hx(i-1,j))/(2*delx)+(Hy(i,j+1)-
Hy(i,j-1))/(2*dely)...
    -(P(i+1,j)+P(i-1,j))/delx^2 - (P(i,j+1)+P(i,j-
1))/dely^2)*...
      (-1/((2/delx^2)+(2/dely^2))) ;

  err = err + abs(Pold(i,j)-P(i,j));

  Pold(i,j)=P(i,j);
    end
end
norm = err/((nx-2)*(ny-2));


end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
err_u = 0;
err_v = 0;
u_new = zeros(nx,ny);
v_new = zeros(nx,ny);

for i = 2:nx
```

```
    for j = 2:ny-1

        u_new(i,j)=u(i,j)+delt*(Hx(i,j)-(P(i+1,j)-P(i-
1,j))/2*delx);

        v_new(i,j)=v(i,j)+delt*(Hy(i,j)-(P(i,j+1)-
P(i,j-1))/2*dely);

        err_u = err_u + abs(u(i,j)-u_new(i,j));
        err_v = err_v + abs(v(i,j)-v_new(i,j));

    end
end

norm_u = err_u/(nx-2)*(ny-2);
norm_v = err_v/(nx-2)*(ny-2);


u(1:nx,1:ny) = u_new(1:nx,1:ny);
u(1,:)= u(nx,:);
v(1:nx,1:ny) = v_new(1:nx,1:ny);
v(1,:)= v(nx,:);
disp(['Time iter ', int2str(time_iter)]);

%%% Energy & Mass%%%%
e = 0;
ut=u';
vt=v';
dsum=0;
for i = 2:nx-1
    for j=2:ny-1
        e= e + (ro * u(i,j)^2);

    end
end

for i=2:ny-1
    for j=2:nx-1
        da = ut(i,j)*(ut(i+1,j)-ut(i-1,j))/(2*delx);
        db = vt(i,j)*(ut(i,j+1)-ut(i,j-1))/(2*dely);
        dc = ut(i,j)*(vt(i+1,j)-vt(i-1,j))/(2*delx);
        dd = vt(i,j)*(vt(i,j+1)-vt(i,j-1))/(2*dely);
```

```
        dsum = dsum + da+db+dc+dd ;
    end
end

%%%%%%%%%%

    E(time_iter,1) = e;
    div(time_iter,1) = dsum;


    if time_iter==490
       u20 = ut;
       save u20.mat;
    elseif time_iter==980
        u40 = ut;
         save u40.mat;
    elseif time_iter==1470
        u60 = ut;
        save u60.mat;
    elseif time_iter==1960
        u80 = ut;
        save u80.mat;
    elseif time_iter==2447
        u100 = ut;
         save u100.mat;
    end


end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% With Momentum %%%%%%%%%%
if (MI == 1)

     P = [P1' ; P];

    for i=1:nx
        for j = 2:ny-1
            u(i,j) = (u(i,j)+(delt*Hx(i,j)))/delx -
(P(i+2,j)-P(i,j))/2 ;
            v(i,j) = (v(i,j)+(delt*Hy(i,j)))/dely ;
        end
    end
```

```
u(nx+1,:)=u(2,:);

%     while (norm > tol)
for k = 1:1000
Piter = Piter+1;
Pold = P;
for i = 3:nx
    for j = 2:ny-1

P(i,j)= ((u(i-2,j)-u(i,j))*2/delx -
P(i+2,j)+2*P(i+1,j)+4*P(i-1,j)+P(i-2,j))/6 ;

  err = err + abs(Pold(i,j)-P(i,j));

  Pold(i,j)=P(i,j);
    end
end
norm = err/((nx-2)*(ny-2));

% P(1:nx,1)=P(1:nx,2);
% P(1:nx,ny)=P(1:nx,ny-1);
disp(['PPE iter ', int2str(Piter)]);
    end


    err_u = 0;
err_v = 0;
u_new = zeros(nx,ny);
v_new = zeros(nx,ny);

for i = 2:nx
    for j = 2:ny-1

        u_new(i,j)=u(i,j)+delt*(Hx(i,j)-(P(i+1,j)-P(i-
1,j))/2*delx);

        v_new(i,j)=v(i,j)+delt*(Hy(i,j)-(P(i,j+1)-
P(i,j-1))/2*dely);

        err_u = err_u + abs(u(i,j)-u_new(i,j));
        err_v = err_v + abs(v(i,j)-v_new(i,j));
```

```
    end
end

norm_u = err_u/(nx-2)*(ny-2);
norm_v = err_v/(nx-2)*(ny-2);


u(1:nx,1:ny) = u_new(1:nx,1:ny);
u(1,:)= u(nx,:);
v(1:nx,1:ny) = v_new(1:nx,1:ny);
v(1,:)= v(nx,:);
disp(['Time iter ', int2str(time_iter)]);

end

end


Pt = P';
save E.mat;
save div.mat;
save ut.mat;
save Pt.mat;
```