Universitat Politècnica de Catalunya
Master of Science in Computational Mechanics
Finite Element in Fluids
FEF
Spring Semester 2017/2018

MatLab Exercises
Incompressible Navier Stokes Equations

Luan Malikoski Vieira

May 30, 2018

Introduction

This report will cover the 2D Stokes and Navier Stokes equations. As proposed by exercises, a Stokes problem with analytical solution will be studied. After, the so-called lid-driven cavity flow will be studied with Stokes and Navier-stokes models. For each problem, Stokes with analytical solution or Cavity flow, a brief explanation on MatLab algorithm changes will be given, followed by presentation of respective results.

1 Exercise 1: Stokes problem with analytical solution

The items of Exercise 1 regarding Stokes flow with analytical solution will be discussed in the following sections separately.

1.1 Item (a): Computation of H^1 and L^2 norm errors

The error norms H^1 and L^2 are computed for the velocity and pressure fields respectively considering the expressions in (1) and (2). Two interpolation combination for Velocity and pressure fields are tested as follows: Q2Q0 and Q2Q1.

$$H^1 = \int_\Omega \left[\left(\frac{\partial u_1^h}{\partial x} - \frac{\partial u_1}{\partial x} \right)^2 + \left(\frac{\partial u_2^h}{\partial y} - \frac{\partial u_2}{\partial y} \right)^2 \right]^{1/2} d\Omega \tag{1}$$

$$L^2 = \int_\Omega \left[\left(p^h - p \right)^2 \right]^{1/2} d\Omega \tag{2}$$

Matlab Routine for H^1 and L^2 norm errors computation.

The function for evalution of the error in Velocity (H^1) and Pressure (L^2) fields has the following input/output form.

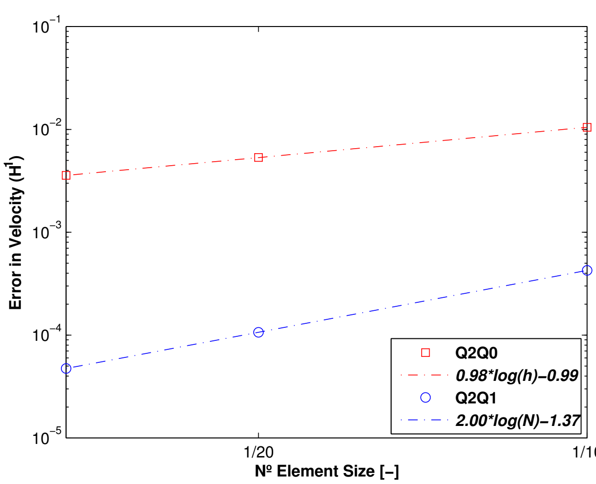
```
function [Ev,Ep] = Errors_Eval(X,T,XP,TP,referenceElement,P,V)
```

The computation of the error is done by integration over the domain Ω in each gauss point. The main code section where errors are evaluated at the gauss point level is found below.

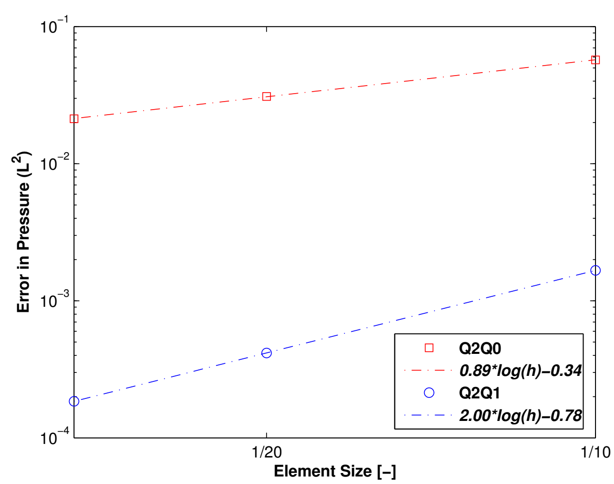
```
for ig = 1:ngaus
    N_ig    = N(ig,:);
    Nxi_ig  = Nxi(ig,:);
    Neta_ig = Neta(ig,:);
    NP_ig = NP(ig,:);
    Jacob = [
        Nxi_ig(1:ngeom)*(Xe(:,1))    Nxi_ig(1:ngeom)*(Xe(:,2))
        Neta_ig(1:ngeom)*(Xe(:,1))   Neta_ig(1:ngeom)*(Xe(:,2))
        ];
    dvolu = wgp(ig)*det(Jacob);
    res = Jacob\[Nxi_ig;Neta_ig];
    nx = res(1,:);
    ny = res(2,:);

    dv1x = nx*Ve(:,1); % DERIVTIVE OF APPROXIMATION u WRT x
    dv2y = ny*Ve(:,2); % DERIVTIVE OF APPROXIMATION u WRT y
    ph = NP_ig*Pe;     % APPROXIMATION P

    x_v = N_ig(1:ngeom)*Xe;
    [~,~,ue_x,~,~,ve_y,~] = ExactSol(x_v);%EXACT SOLUTION u AT GAUSS POINT
    Ev = Ev + sqrt((dv1x - ue_x)^2 + (dv2y - ve_y)^2)*dvolu; % H1 NORM

    x_p = NP_ig(1:nenP)*Xp;
    [~,~,~,~,~,~,pe] = ExactSol(x_p); %EXACT SOLUTION P AT GAUSS POINT
    Ep = Ep + sqrt((ph - pe)^2)*dvolu; % L2 NORM
end
```

Results: H^1 and L^2 norm errors.

The error norm for velocity and pressure are shown in Figure (1). The adjusted curve for each case is also presented along with its equation. It can be observed that for the Q2Q0 formulation, both velocity and pressure error norms presents a slop near of 1 (0.98 and 0.89 respectively). For the Q2Q1 case the slop becomes 2 for both norm errors. This is expected as the error norm order of the two variables is controlled by the lowest order one. The theoretical norm error orders for the case of the Q2Q0 are 1 for H^1 norm and 2 for L^2 norm, while for Q2Q1 both H^1 and L^2 orders are 2.



(a)



(b)

Figure 1: Error Norms: (a) Velocity (H^1);(b) Pressure (L^2).

1.2 Item (b): Stabilized formulation for P1P1 interpolation

The stabilized Galerkin formulation, by a Least Square approach (GLS), of the stokes problem for the case of P1P1 interpolation is found in equations (3) and (4). Where $\tau_e = 1/3$ is used as an optimal value for bilinear quadrilateral elements.

$$a(w^h, v^h) + b(w^h, p^h) = (w^h, b^h) + (w^h, t^h)_{\Gamma_N} - a(w^h, v^h_D) \tag{3}$$

$$b(b^h, q^h) - \sum_{e=1}^{n_{el}} \tau_e (\nabla q^h, \nabla p^h)_{\Omega_e} = -b(v^h_D, q^h) - \sum_{e=1}^{n_{el}} \tau_e (\nabla q^h, b^h)_{\Omega_e} \tag{4}$$

As linear elements are used for both velocity and pressure interpolation, the weak form of momentum equation is not changed. Only the compressibility constrains is changed. Thus, two new tensors are added to the problem corresponding to the two terms in the equation (4), named here as A_e and h_e.

Consequently the system of equations to be solved becomes:

$$\begin{bmatrix} K & G^T \\ G & A_e \end{bmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ h + h_e \end{pmatrix} \tag{5}$$

Matlab Routine for P1P1 Galerkin stabilized formulation.

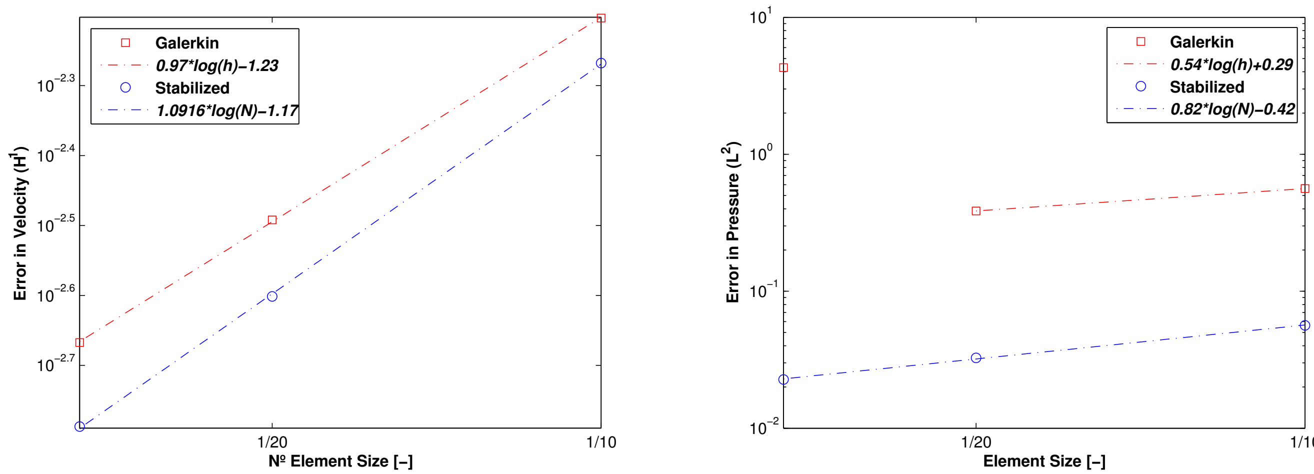
The code section, at gauss point level, needed to the evaluation of the matrix A_e and vector h_e are shown below.

```
for ig = 1:ngaus
    N_ig    = N(ig,:);
    Nxi_ig  = Nxi(ig,:);
    Neta_ig = Neta(ig,:);
    NP_ig = NP(ig,:);
    NPxi_ig = NPxi(ig,:);
    NPeta_ig = NPeta(ig,:);
    Jacob = [
        Nxi_ig(1:ngeom)*(Xe(:,1))   Nxi_ig(1:ngeom)*(Xe(:,2))
        Neta_ig(1:ngeom)*(Xe(:,1))  Neta_ig(1:ngeom)*(Xe(:,2))
        ];
    JacobP =[
        NPxi_ig(1:nedofP)*(Xp(:,1)) NPxi_ig(1:nedofP)*(Xp(:,2))
        NPeta_ig(1:nedofP)*(Xp(:,1))    NPeta_ig(1:nedofP)*(Xp(:,2))
        ];
    dvolu = wgp(ig)*det(Jacob);
    res = Jacob\[Nxi_ig;Neta_ig];
    nx = res(1,:);
    ny = res(2,:);
    resP = JacobP\[NPxi_ig;NPeta_ig]; %DERIVATIVES OF P SHAPE FUNCTION
    Nx_P = resP(1,:); %DERIVATIVE OF P SHAPE FUNCTIONS WRT TO x
    Ny_P = resP(2,:); %DERIVATIVE OF P SHAPE FUNCTIONS WRT TO y
    Ngp = [reshape([1;0]*N_ig,1,nedofV); reshape([0;1]*N_ig,1,nedofV)];
    Nx = [reshape([1;0]*nx,1,nedofV); reshape([0;1]*nx,1,nedofV)];
    Ny = [reshape([1;0]*ny,1,nedofV); reshape([0;1]*ny,1,nedofV)];
    dN = reshape(res,1,nedofV);
```

```
    Ke = Ke + (Nx'*Nx+Ny'*Ny)*dvolu;
    Ge = Ge - NP_ig'*dN*dvolu;
    Ae = Ae - (Nx_P'*Nx_P + Ny_P'*Ny_P)*dvolu; %LEFT HAND SIDE EST. TERM
    x_ig = N_ig(1:ngeom)*Xe;
    f_igaus = SourceTerm(x_ig);
    fe = fe + Ngp'*f_igaus*dvolu;
    he = he - resP'*f_igaus*dvolu; %RIGHT HAND SIDE ESTABILAZATION TERM
end
```

Results: Stabilized Formulation P1P1.

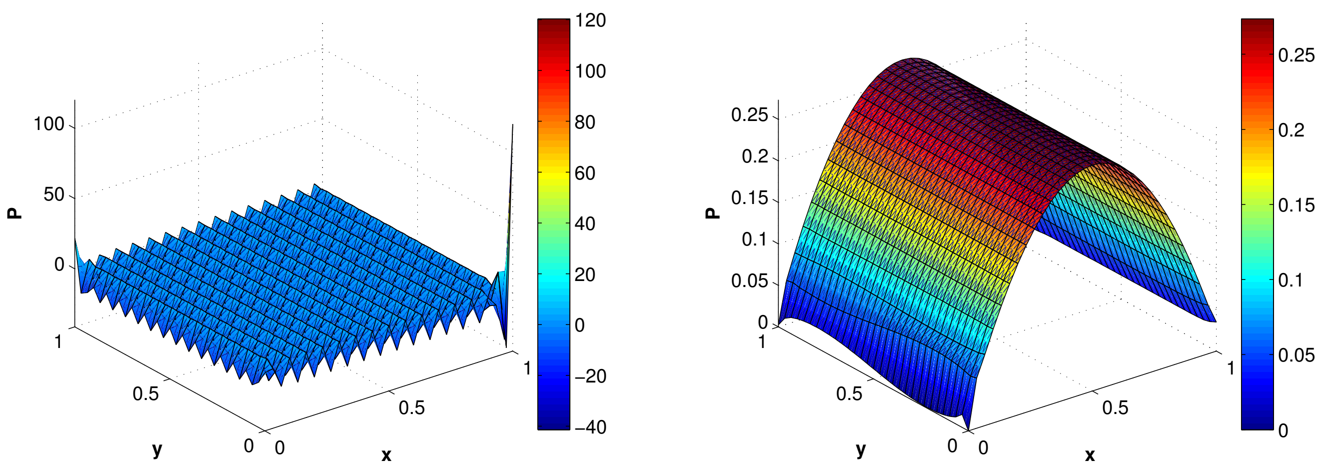
The error norm for velocity and pressure are shown in Figure (2). The adjusted curve for each case is also presented along with its equation. For the Galerkin case the H^1 error norm order in velocity is near 1 (0.97) while the L^2 error norm order for the pressure field do not present a smooth slope due to pressure oscillations in this unstable case, as shown in Figure (3) (a). For the stabilized formulation, both error norms, H^1 and L^2 presents a slop near 1 (1.096 and 0.82 respectively).The theoretical norm error orders for the case of the P1P1 are 1 for both H^1 and L^2, which is accordance with the found values.



(a) (b)

Figure 2: Error Norms: (a) Velocity (H^1);(b) Pressure (L^2).

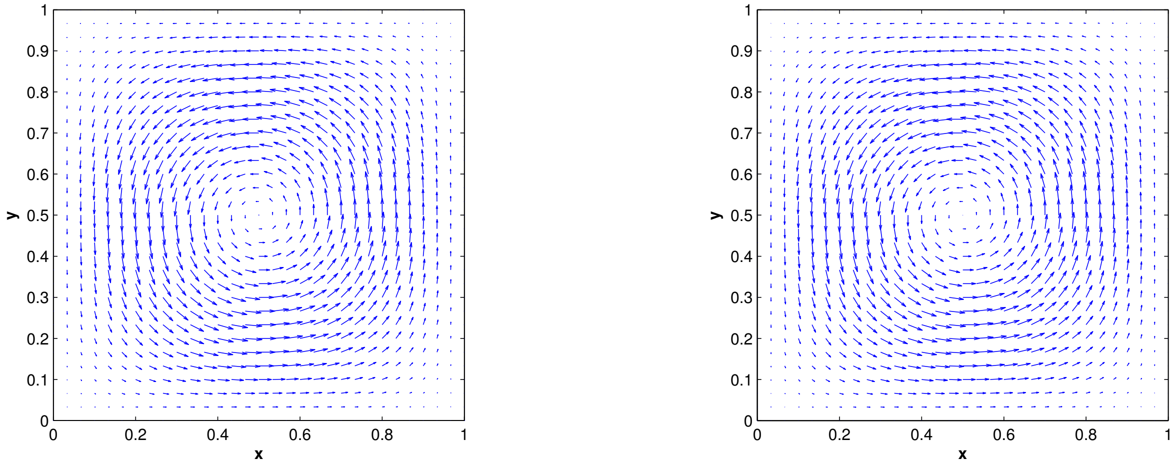
Results for the Pressure field for both Galerkin and stabilized formulation are shown in Figure (3), for a 30x30 uniform mesh. Here pressure oscillations are vanished when stabilization is applied and pressure field resembles the analytic one.



(a) (b)

Figure 3: Pressure Field - 30x30 P1P1 mesh:(a) Galerkin ;(b) Stabilized.

Results for the velocity vector field for both Galerkin and stabilized formulation are shown in Figure (4), for a 30x30 uniform mesh. Here results are similar for both formulations, as also seen in the H^1 error plot of figure (2) (a).



(a) (b)

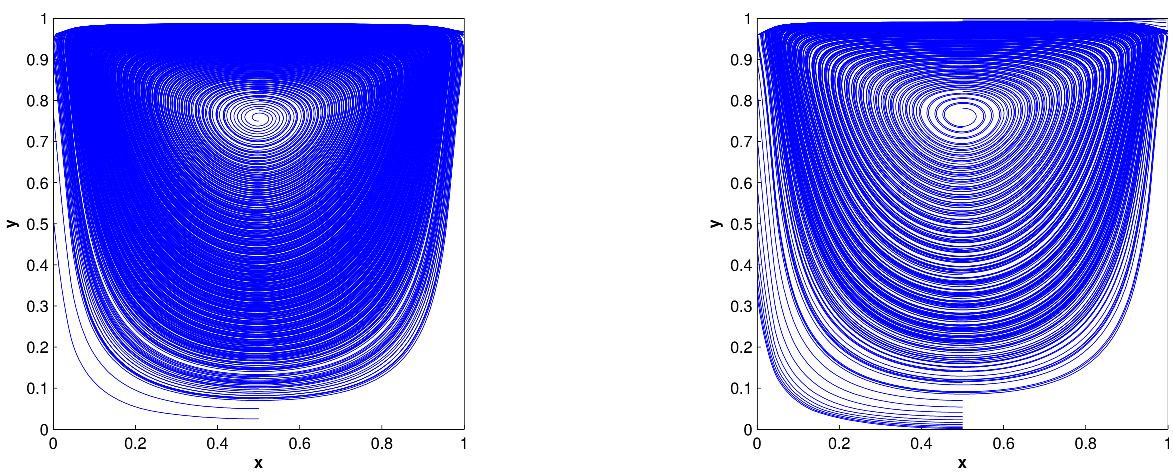
Figure 4: Velocity Vector Field - 30x30 P1P1 mesh:(a) Galerkin ;(b) Stabilized.

2 Exercise 2: Cavity Problem

The items of Exercise 1 regarding Stokes flow with analytical solution will be discussed in the following sections separately.

2.1 Item (a): Stokes solution of Cavity problem

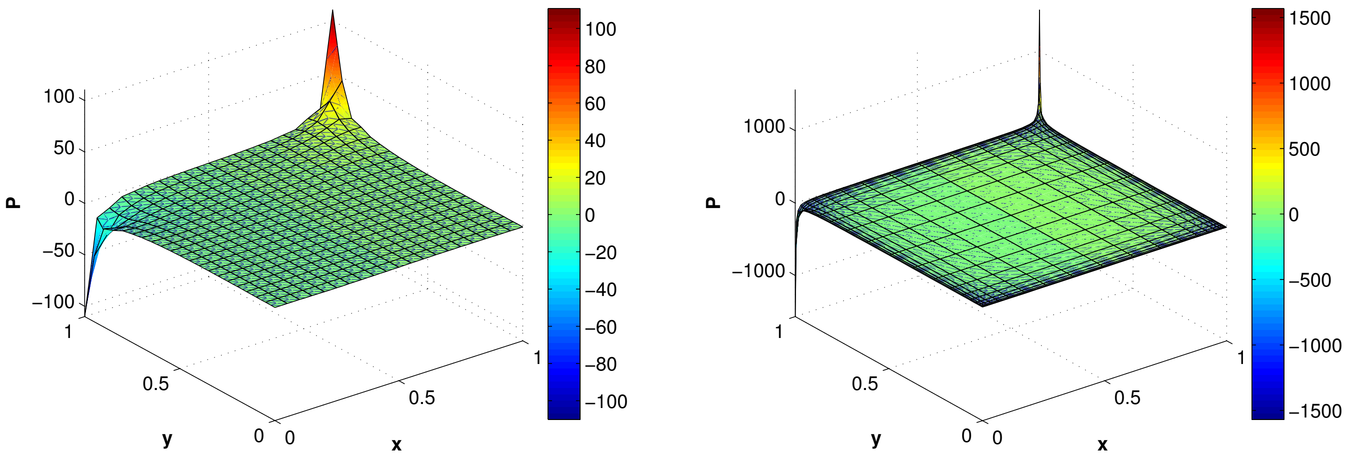
Velocity field results are not too affected by changing the mesh from a uniform to a adaptive one, refined at the walls. It can be observed in Figure (5).



(a) (b)

Figure 5: Stream lines - 20x20 Q2P1 mesh:(a) Uniform ;(b) Adaptive.

The pressure field, however, has a significant change when an adaptive mesh is employed. As seen in Figure (6), the pressure discontinuity at the extremes nodes of the Dirichlet boundary, due to discontinuous velocity boundary condition, is better captured. As discontinuity is present in a smaller space, for the adaptive mesh case, pressure peak is increased to conserve energy (FEM seen as an energy balance).



(a) (b)

Figure 6: Pressure Field - 20x20 Q2P1 mesh:(a) Uniform ;(b) Adaptive.

2.2 Items (b) and (c): Navier - Stokes solution of Cavity problem

The non-linearity of the Navier - Stokes model of the Cavity problem is solved by Picard and Newton - Rapshon methods. In order to implement those methods the convective, and nonlinear, term of the weak form is discretized in two ways. This leads to the definition of C_1 and C_2 as follows. Where $\mathbf{a}$ is an approximation for $\mathbf{u}$.

$$\int_\Omega \mathbf{w}(\mathbf{a}\cdot\nabla)\mathbf{u}d\Omega = C(\mathbf{a})\mathbf{u} = \mathbf{C_1 u} \tag{6}$$

$$\int_\Omega \mathbf{w}(\mathbf{u}\cdot\nabla)\mathbf{a}d\Omega = C(\mathbf{u})\mathbf{a} = \mathbf{C_2 a} \tag{7}$$

Matlab Routine for evaluation of Convective matrix C_1.

```
function Ce = EleMatStokes(Xe,ngeom,nedofV,ngaus,wgp,N,Nxi,Neta,Conve)

Ce = zeros(nedofV,nedofV);
% Loop on Gauss points
for ig = 1:ngaus
    N_ig    = N(ig,:);
    Nxi_ig  = Nxi(ig,:);
    Neta_ig = Neta(ig,:);
    Jacob = [
        Nxi_ig(1:ngeom)*(Xe(:,1))    Nxi_ig(1:ngeom)*(Xe(:,2))
        Neta_ig(1:ngeom)*(Xe(:,1))   Neta_ig(1:ngeom)*(Xe(:,2))
        ];
    dvolu = wgp(ig)*det(Jacob);
    res = Jacob\[Nxi_ig;Neta_ig];
    nx = res(1,:);
    ny = res(2,:);
    % Convective velocity
    Conv = N_ig*Conve;
    vx = Conv(1); % VELOCITY IN GAUSS POINT x DIRECTION
    vy = Conv(2); % VELOCITY IN GAUSS POINT y DIRECTION
    % Gradient
    Nx = [reshape([1;0]*nx,1,nedofV); reshape([0;1]*nx,1,nedofV)];
    Ny = [reshape([1;0]*ny,1,nedofV); reshape([0;1]*ny,1,nedofV)];
    Ni = [reshape([1;0]*N_ig,1,nedofV); reshape([0;1]*N_ig,1,nedofV)];

    Ce = Ce + ((Ni')*vx*Nx+(Ni')*vy*Ny)*dvolu; % CONVECTIVE MATRIX

end
```

Matlab Routine for evaluation of Convective matrix C_2.

```
function Ce = EleMatStokes(Xe,ngeom,nedofV,ngaus,wgp,N,Nxi,Neta,Conve)

Ce = zeros(nedofV,nedofV);
% Loop on Gauss points
for ig = 1:ngaus
    N_ig    = N(ig,:);
    Nxi_ig  = Nxi(ig,:);
    Neta_ig = Neta(ig,:);
    Jacob = [
        Nxi_ig(1:ngeom)*(Xe(:,1))    Nxi_ig(1:ngeom)*(Xe(:,2))
        Neta_ig(1:ngeom)*(Xe(:,1))  Neta_ig(1:ngeom)*(Xe(:,2))
        ];
    dvolu = wgp(ig)*det(Jacob);
    res = Jacob\[Nxi_ig;Neta_ig];
    % Convective velocity
    dConv = res*Conve; % DERIVATIVE OF VELOCITY IN GAUSS POINT
    Ni = [reshape([1;0]*N_ig,1,nedofV); reshape([0;1]*N_ig,1,nedofV)];
    Ce = Ce + Ni'*dConv'*Ni*dvolu;
end
```

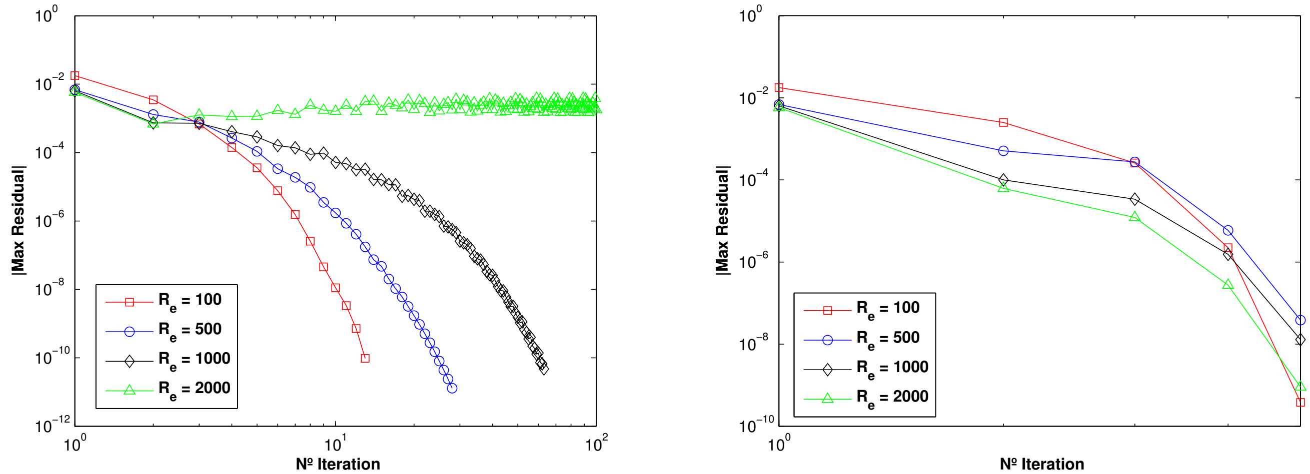
Matlab Routine for Newton - Raphson method.

```
iter = 0; tol = 0.5e-08;
while iter < 100
    fprintf('Iteration = %d\n',iter);

    C1 = ConvectionMatrix(X,T,referenceElement,velo);
    C2 = ConvectionMatrix2(X,T,referenceElement,velo);
    Cred1 = C1(dofUnk,dofUnk);
    Cred2 = C2(dofUnk,dofUnk);

    Atot = A;
    Atot(1:nunkV,1:nunkV) = A(1:nunkV,1:nunkV) + Cred1;
    btot = [fred - C1(dofUnk,dofDir)*valDir; zeros(nunkP,1)];

    % Computation of residual
    res = Atot*sol0 - btot ;
    % Computation of velocity and pressure
    %Jacobian Matrix
    J11 = Kred + Cred1 + Cred2;
    J = [ J11     Gred'
          Gred    zeros(nunkP)];
    % Solution Increment
    solInc = -J\res;

    % Update the solution
    veloInc = zeros(ndofV,1);
    veloInc(dofUnk) = solInc(1:nunkV);
    presInc = solInc(nunkV+1:end);
    velo = velo + reshape(veloInc,2,[])';
    pres = pres + presInc;
```

```
    % Check convergence
    delta1 = max(abs(veloInc));
    delta2 = max(abs(res));
    fprintf('Velocity increment=%8.6e, Residue max=%8.6e\n',delta1,delta2);
    if delta1 < tol*max(max(abs(velo))) && delta2 < tol
        fprintf('\nConvergence achieved in iteration number %g\n',iter);
        break
    end

    % Update variables for next iteration
    veloVect = reshape(velo',ndofV,1);
    sol0 = [veloVect(dofUnk); pres];
    iter = iter + 1;
end
```

Results: Picard and Newton - Raphson method.

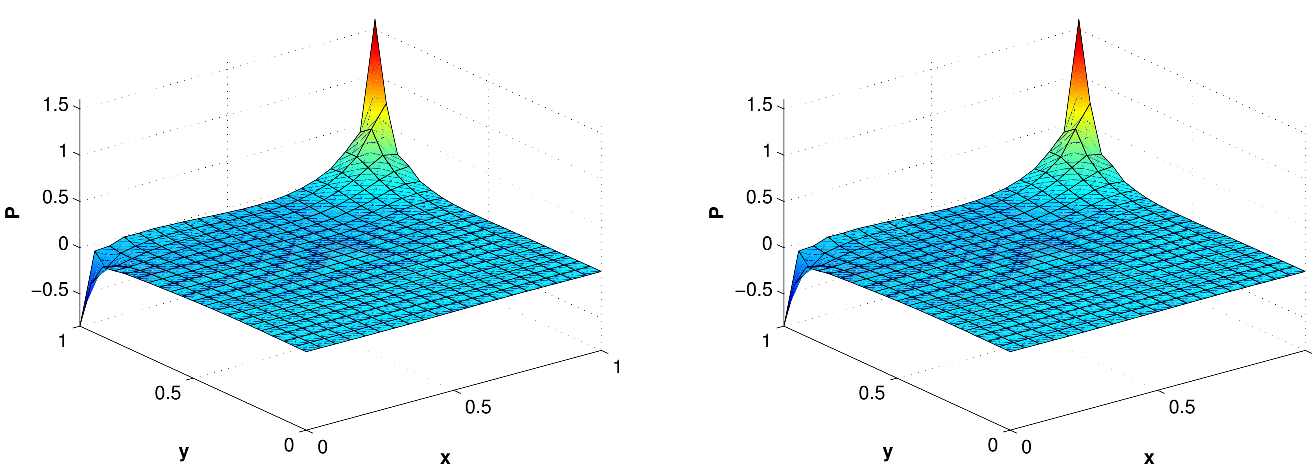
The problem were run using both Picard and Newt-Raphson method to overcome the convective non-linearity. For both methods four Reynolds number were tested (100, 500, 1000, and 2000). The initial condition for the case where $R_e = 100$ was the zero velocity field ($u_x = u_y = 0$) while for the other reynolds numbers, the previous reynolds solution were used to initialize the velocity field (e.g, for $R_e = 500$, $R_e = 100$ solution was the initial condition and so on). Results for the convergence on the Maximum Residual is shown in figure (7). It can be noticed that Picard requires higher iterations to reach the convergence criteria. Also, the higher the R_e, the higher the iterations number. When $R_e = 2000$ Picard can not achieve the convergence to the required level. When using this initial condition scheme, Newton - Raphson performs really well requiring less than 10 iterations to converge for all cases.



(a) (b)

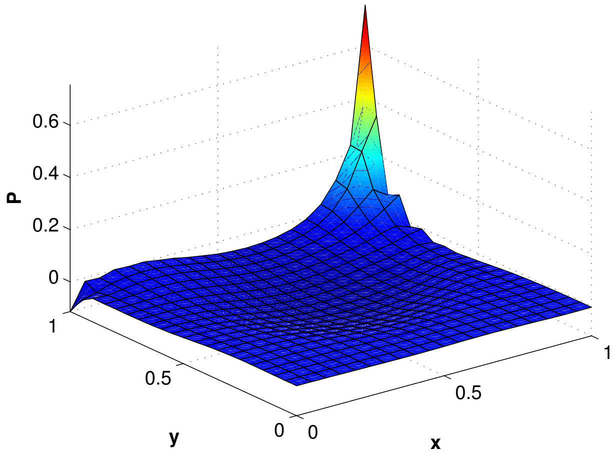
Figure 7: Convergence in Max-Residual - 20x20 Q2P1 mesh:(a) Picard ;(b) Newton - Raphson.

Results for the Pressure and Velocity field for each reynolds number (100, 500, 1000 and 2000) are shown from Figure (8) to Figure (15). As a general result, as the Reynolds number is increased the vortex core is moved downwards to the center of the cavity. Also the vortex strength is increased, as expected. Given this increment in the rotational velocity, the pressure is reduced to conserve the energy (FEM model seen as an energy balance). It can be noticed that near the vortex core the pressure reduction is higher and a pressure valley is formed in the graph..

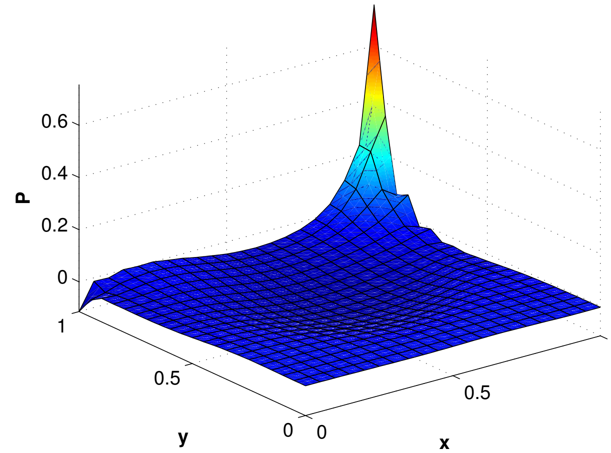


(a) (b)

Figure 8: Pressure Field - 20x20 Q2P1 mesh $R_e = 100$:(a) Picard ;(b) Newton - Raphson.

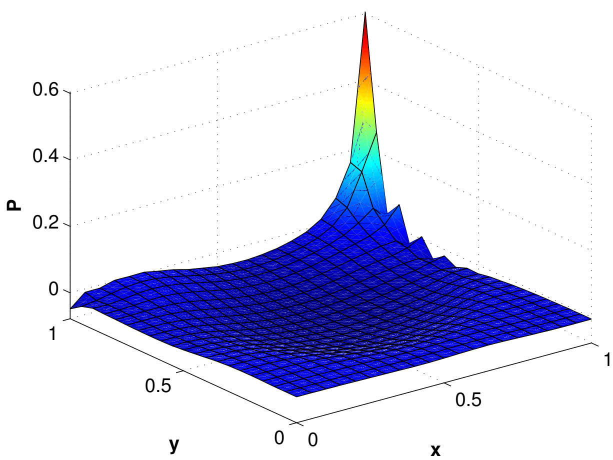


(a)

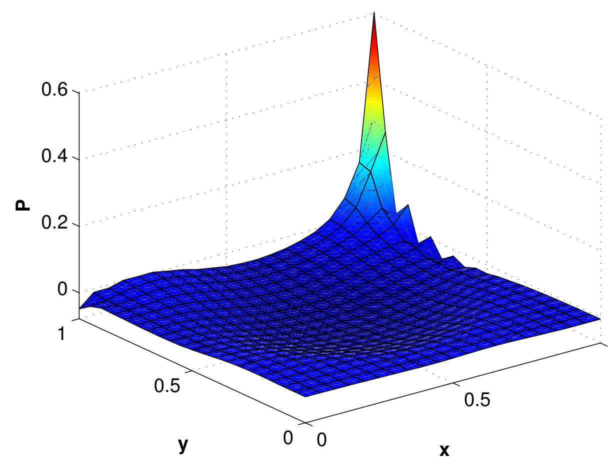


(b)

Figure 9: Pressure Field - 20x20 Q2P1 mesh $R_e = 500$:(a) Picard ;(b) Newton - Raphson.

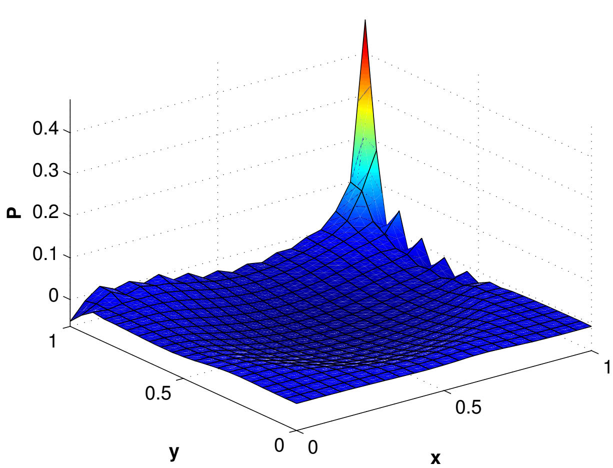


(a)

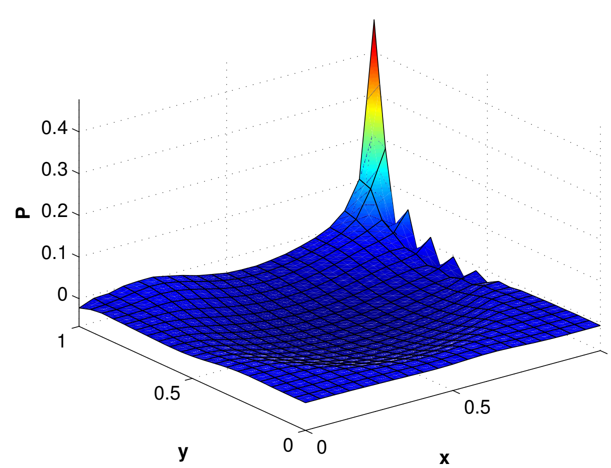


(b)

Figure 10: Pressure Field - 20x20 Q2P1 mesh $R_e = 1000$:(a) Picard ;(b) Newton - Raphson.

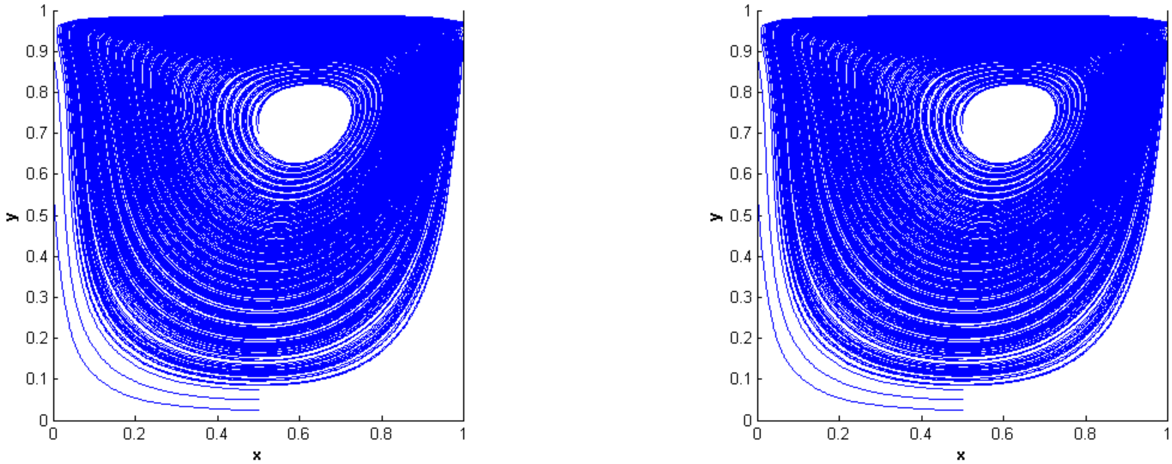


(a)



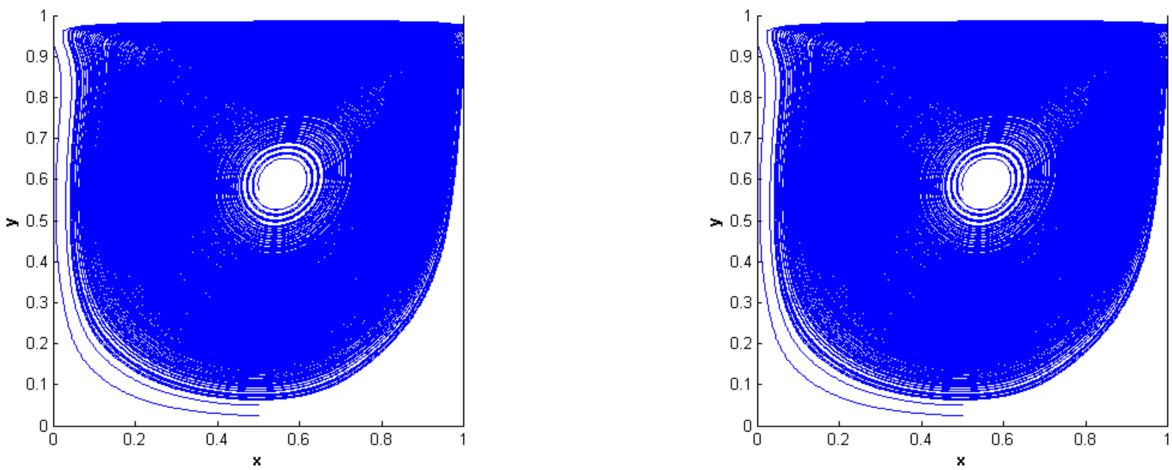
(b)

Figure 11: Pressure Field - 20x20 Q2P1 mesh $R_e = 2000$:(a) Picard ;(b) Newton - Raphson.



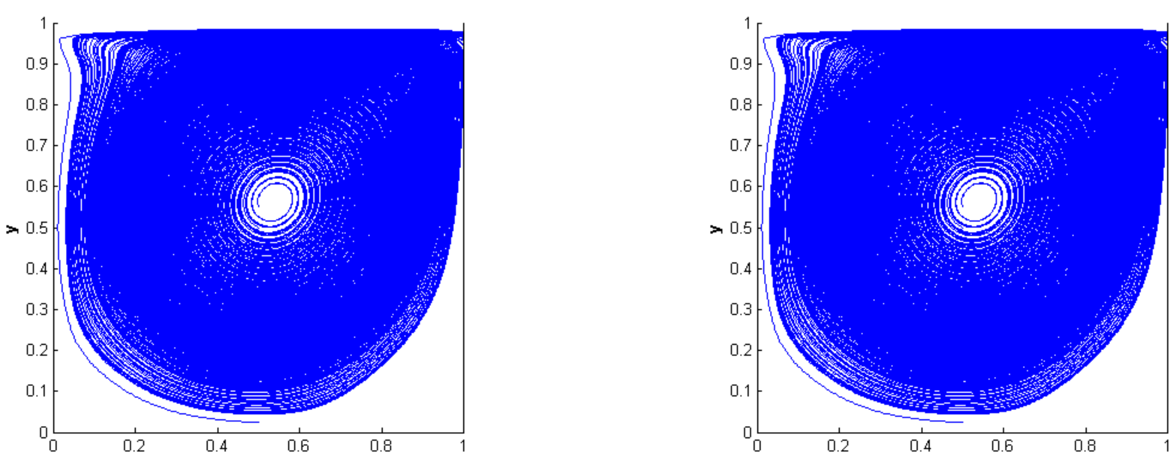
(a) (b)

Figure 12: Streamlines - 20x20 Q2P1 mesh $R_e = 100$:(a) Picard ;(b) Newton - Raphson.



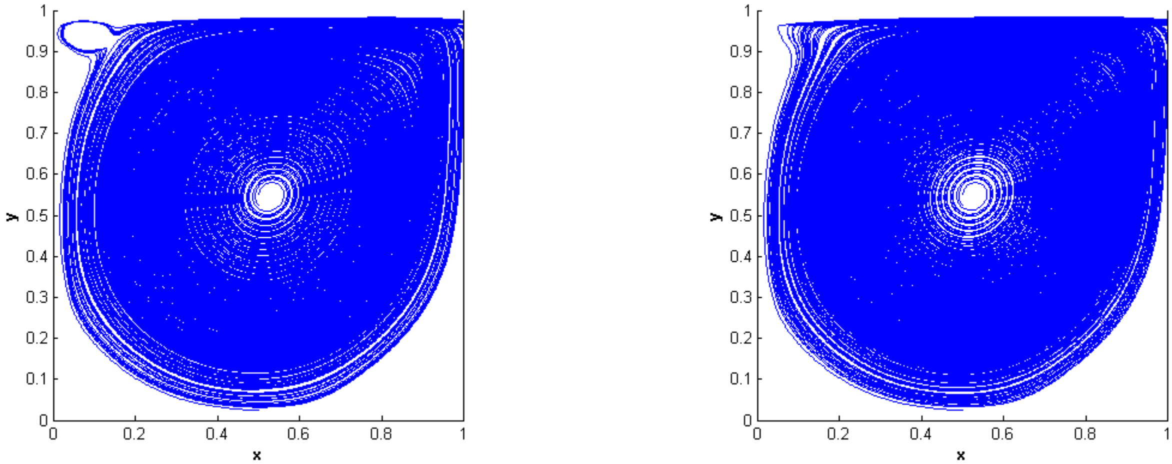
(a) (b)

Figure 13: Streamlines - 20x20 Q2P1 mesh $R_e = 500$:(a) Picard ;(b) Newton - Raphson.



(a) (b)

Figure 14: Streamlines - 20x20 Q2P1 mesh $R_e = 1000$:(a) Picard ;(b) Newton - Raphson.



(a)

(b)

Figure 15: Streamlines - 20x20 Q2P1 mesh $R_e = 2000$:(a) Picard ;(b) Newton - Raphson.