Finite Elements in Fluids, Assignment on Incompressible Viscous Flows

Jose Raul Bravo Martinez, MSc Computational Mechanics

April 3, 2019

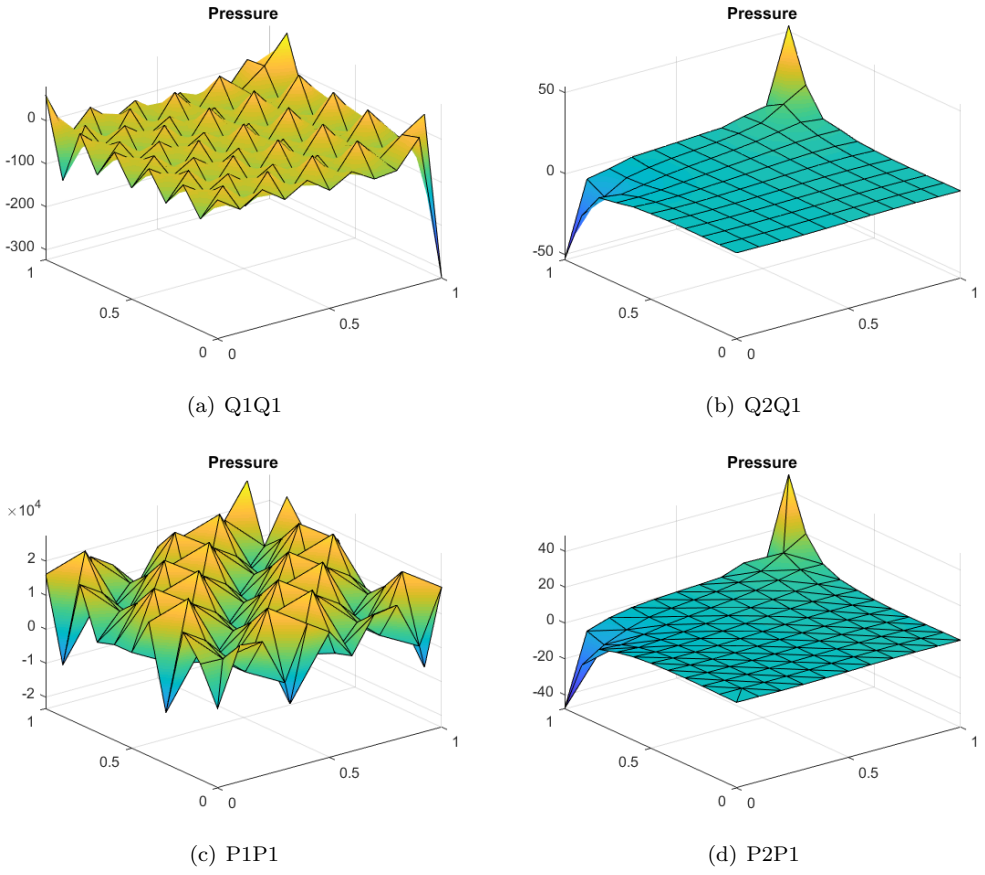
On Stokes Equations:

- **Describe physically the problem based on the values of the BC**

  The cavity problem in 2D is a representation of a square filled with fluid. The left, right and bottom edges are no slip walls, The condition prescribed on the upper part is velocity in the right direction, and no normal velocity is allowed. This is the one condition that puts the fluid in motion.

- **Solve the problem using different element types: • Q1Q1, Q2Q1 • P1P1, P2P1 Which ones are stable?**

  The code, as is, provides the results shown in figure 1, when running for the mentioned elements.



(a) Q1Q1 (b) Q2Q1

(c) P1P1 (d) P2P1

Figure 1: Results of the simulation before modifications

It can be seen that those elements, which higher order on velocity than pressure are stable.

Actually, this is not a sufficient condition. One should also pay attention to comply with the LBB condition, and Q2Q1, and P2P1 elements do comply with it.

- **For the unstable elements, program a stabilization method. Write a report describing the discretization of the method and the results obtained so far**

  Starting from the weak form of the problem:

$$\begin{cases} \int_\Omega \nabla \boldsymbol{w} : \nu \nabla \boldsymbol{v} d\Omega - \int_\Omega p \nabla \cdot \boldsymbol{w} d\Omega = \int_\Omega \boldsymbol{w} \cdot \boldsymbol{b} d\Omega & \forall \boldsymbol{w} \in \mathcal{V} \\ \int_\Omega q \nabla \cdot \boldsymbol{v} d\Omega = 0 & \forall \boldsymbol{q} \in \mathcal{Q} \end{cases}$$

  One obtains the following matrix system:

$$\begin{bmatrix} \boldsymbol{K} & \boldsymbol{G} \\ \boldsymbol{G^T} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{p} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{0} \end{bmatrix}$$

  Following GLS stabilization technique, two terms must be added (For linear elements).

$$\begin{bmatrix} \boldsymbol{K} & \boldsymbol{G} \\ \boldsymbol{G^T} & \bar{\boldsymbol{L}} \end{bmatrix} \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{p} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \bar{\boldsymbol{f}}_{\boldsymbol{q}} \end{bmatrix}$$

  Where:

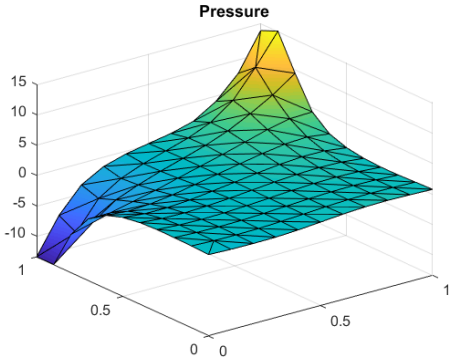
$$\bar{\boldsymbol{L}} = \sum_e \int_{\Omega^e} \tau_1 (\nabla q) \cdot (\nabla p) d\Omega^e \qquad \bar{\boldsymbol{f}}_{\boldsymbol{q}} = \sum_e \int_{\Omega^e} \tau_1 (\nabla q) \cdot (-\boldsymbol{f}) d\Omega^e$$

  This was implemented in the code as shown in figure 2.

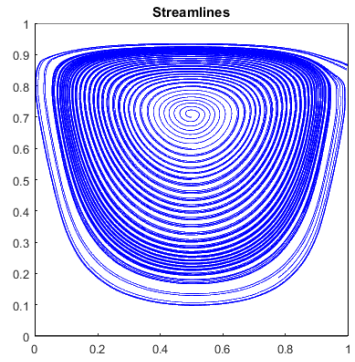
```
    Ngp = [reshape([1;0]*N_ig,1,nedofV); reshape([0;1]*N_ig,1,nedofV)];
    % Gradient
    Nx = [reshape([1;0]*nx,1,nedofV); reshape([0;1]*nx,1,nedofV)];
    Ny = [reshape([1;0]*ny,1,nedofV); reshape([0;1]*ny,1,nedofV)];
    % Divergence
    dN = reshape(res,1,nedofV);

    Ke = Ke + (Nx'*Nx+Ny'*Ny)*dvolu;
    Ge = Ge - NP_ig'*dN*dvolu;
    x_ig = N_ig(1:ngeom)*Xe;
    f_igaus = SourceTerm(x_ig);
    fqe= fqe + res'*f_igaus*dvolu;
    Le=  Le + tau*(nx'*nx + ny'*ny)*dvolu;
    fe = fe + Ngp'*f_igaus*dvolu;
end
```

Figure 2: Modifications to the Code for Stabilization

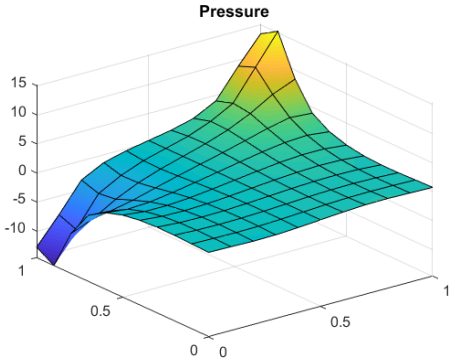
After performing the modifications, the two elements which did not comply with the LBB condition, were tested, and the plots obtained are shown in figure 3.



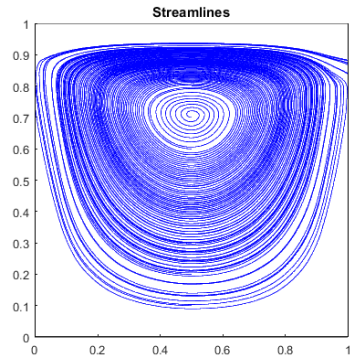
(a) P1P1



(b) P1P1



(c) Q1Q1



(d) Q1Q1

Figure 3: Results of the simulation after modifications

On Navier-Stokes Equations

- **Complete the Piccard's method with the convective term C(v).**

The code provided solves the Navier Stokes Equations:

$$\begin{cases} a(\boldsymbol{w},\boldsymbol{v}) + c(\boldsymbol{w},\boldsymbol{v},\boldsymbol{v}) + b(\boldsymbol{w},p) = (\boldsymbol{w},\boldsymbol{b}) + (\boldsymbol{w},t)_{\Gamma N} & \forall \boldsymbol{w} \in \mathcal{V} \\ b(\boldsymbol{v},q) = 0 & \forall \boldsymbol{q} \in \mathcal{Q} \end{cases}$$

Which gives the following system of equations:

$$\begin{bmatrix} \boldsymbol{K} + \boldsymbol{C}(\boldsymbol{v}) & \boldsymbol{G} \\ \boldsymbol{G}^{\boldsymbol{T}} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{p} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{0} \end{bmatrix}$$

To add the convection matrix, one needs to add a function to the code. A part of it is shown in figure 5.

```
        % Calculationg the 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)];
        %Velocity in the Gauss point
        V_ig = N_ig*Ve;
        Vx = [V_ig(1) 0;0 V_ig(1)];
        Vy = [V_ig(2) 0;0 V_ig(2)];
        %Convection Matrix
        Ce = Ce + Ngp'*(Vx*Nx+Vy*Ny)*dvolu;
    end
    % Assembly
    C(Te_dof, Te_dof) = C(Te_dof, Te_dof) + Ce;
end
```

Figure 4: Convection Matrix

It can be obsered that Piccard method has a linear convergence speed as seen in figure 5. Which could be improved by applying the Newton-Raphson method. Due to time constraints, Newton-Raphson will be presented in the next assignment.

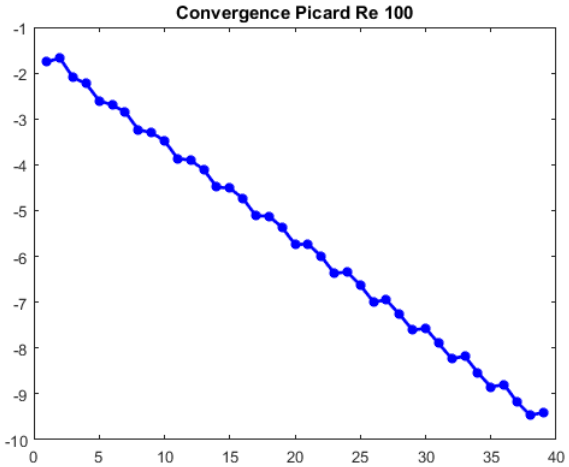


Figure 5: Convergence of Piccard Method