FINITE ELEMENTS IN FLUIDS

Homework 4

Stokes & Navier-Stokes problem

Author: Cristina García Albela

MsC in Computational Mechanics

STOKES EQUATIONS – CAVITY FLOW PROBLEM

Stokes equations are obtained when Navier – Stokes is applied to a highly viscous flow; thus, the convective terms can be neglected if they are compared with the dominant viscous terms. If time dependency is neglected too, steady Stokes problem can be studied formulated in terms of velocity and pressure.

$$\begin{aligned} -\nu\nabla^2 v + \nabla p = b \qquad & in\ \Omega \\ \nabla\cdot v = 0 \qquad & in\ \Omega \\ v = v_D \qquad & on\ \partial\Omega \end{aligned}$$

Cavity problem is going to be studied, being one of the most typical examples for incompressible problems. It models a plane flow of an isothermal fluid in a square lid-driven cavity where Dirichlet BC are defined in each side, implying that pressure is known up to a constant.
Talking about velocity, the upper side moves in its own plane with unit speed, whereas the other remains fixed; appearing a discontinuity in the upper corner nodes ($v_1 = 1$ and $v_1 = 0$). In terms of pressure, at any arbitrary point the left lower corner of the cavity will have $p = 0$ value prescribed and a singulary in the upper corners is introduced by velocity discontinuity.

Discretization of Stokes equations

Applying WRM to Stokes equations and after some algebra (integration by parts and boundary conditions), Stokes equations can be written as

$$\int_\Omega (\nabla\boldsymbol{w}):(\nu\nabla\boldsymbol{v})d\Omega - \int_\Omega (\nabla\cdot\boldsymbol{w})p d\Omega = \int_\Omega \boldsymbol{w}\cdot\boldsymbol{f} d\Omega$$

$$\int_\Omega q\nabla\cdot\boldsymbol{v} d\Omega = 0$$

where $\boldsymbol{w}$ and q are the weighted functions.
Writing the approximation for velocity and pressure together with Galerkin formulation for their weighted functions

$$\boldsymbol{v}(x) \cong \boldsymbol{v^h}(x) = \sum_{j=1}^{n} v_j N_j(x) \qquad\qquad p(x) \cong p^h(x) = \sum_{j=1}^{n} p_j \widehat{N}_j(x)$$

$$w(x) \cong w^h(x) = \sum_{i=1}^{n} N_i(x) w_i \qquad\qquad q \cong q^h = \sum_{i=1}^{n} \widehat{N_i}(x) q_i$$

Introducing these expressions in the previous equation and after some algebra the matrix problem governing Stokes flow is derived, obtaining the following matrix system of equations

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

Implemented the problem in a Matlab code (Figure 1), four different finite element spaces are going to be solved using a 10 elements mesh in each direction and viscosity equal to the unity.

```
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);
fe = fe + Ngp'*f_igaus*dvolu;
```

Figure 1. Element matrix components definition

LBB Condition

Defining the positive symmetric matrix $(\boldsymbol{G}^T\boldsymbol{K}^{-1}\boldsymbol{G})$, it is positive define if $\ker \boldsymbol{G} = \{\boldsymbol{0}\}$. If this is the case the velocity and pressure fields obtained solving the system will be unique; if not, the velocity filed might be stable and convergent, but pressure field will be no stable, presenting oscillatory results. To guarantee the stability of the method LBB compatibility condition has been determined. The LBB condition states that velocity and pressure must have a link between them instead of being chosen arbitrary. To obtain uniquely v and p from the system of equations, a necessary but not sufficient condition is that

$$dimQ^h \leq dimV^h$$

If velocity and pressure discrete spaces satisfy the LBB condition $\ker \boldsymbol{G} = \{\boldsymbol{0}\}$, guaranteeing the existence and uniqueness of the solution.

Stabilization of Stokes problem

In order to obtain better results using velocity – pressure pairs which doesn't satisfy LBB condition (P1P1 and Q1Q1) a stabilization technique, in this case GLS formulation, will be introduced to solve incompressible flow problems.
Now the problem can be written as

$$\int_\Omega \begin{bmatrix} \boldsymbol{W} \\ q \end{bmatrix} \cdot (\boldsymbol{\mathcal{L}}(\boldsymbol{v},p) - \boldsymbol{F}) d\Omega + \sum_e \int_{\Omega_e} \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \boldsymbol{\mathcal{L}}(\boldsymbol{w},q) \cdot (\boldsymbol{\mathcal{L}}(\boldsymbol{v},p) - \boldsymbol{F}) d\Omega = \boldsymbol{0}$$

$$\tau_1 = \alpha_0 \frac{h^2}{4\nu} \qquad \tau_2 = 0 \qquad (\alpha_0 = \frac{1}{3}\ for\ linear\ elements)$$

Working with linear elements the second order derivatives vanish so that the GLS does not affect the momentum equation at all together with τ_2 value, the GLS formulation reduces to

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

Implemented the new system of equations in a Matlab code (Figure 2 & 3), P1P1 and Q1Q1 elements discretization are going to be solved using a 10 and 30 elements mesh in each direction and viscosity equal to the unity.

```
K = zeros(ndofV,ndofV);
G = zeros(ndofP,ndofV);
L = zeros(ndofP,ndofP);
f = zeros(ndofV,1);
fq = zeros(ndofP,1);
```

Figure 2. Element matrix components

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

Figure 3. Element matrix components definition

Results and conclusions – Non stabilization term

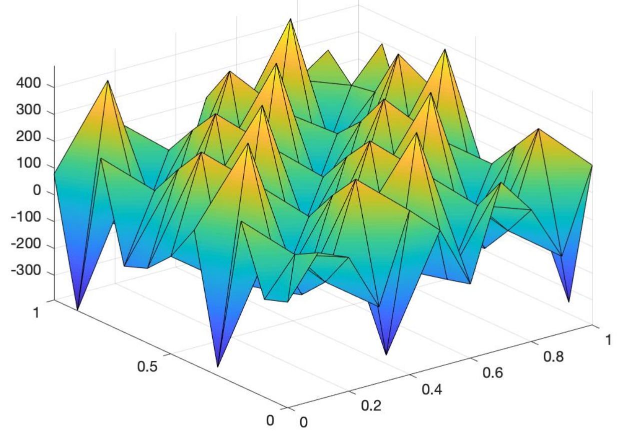

Figure 4. Pressure field for P1P1 elements

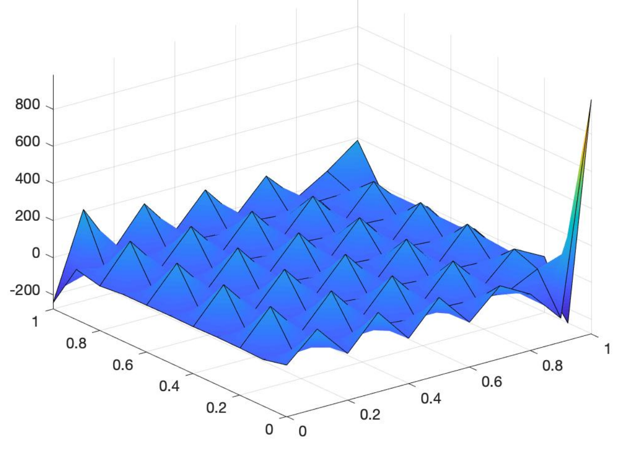

Figure 5. Pressure field for Q1Q1 elements

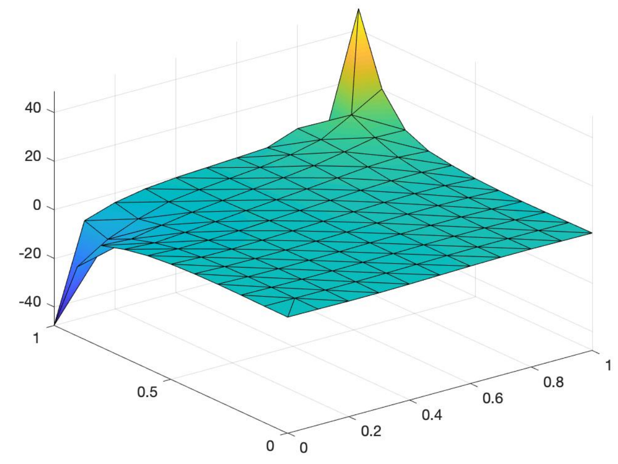

Figure 6. Pressure field for P2P1 elements

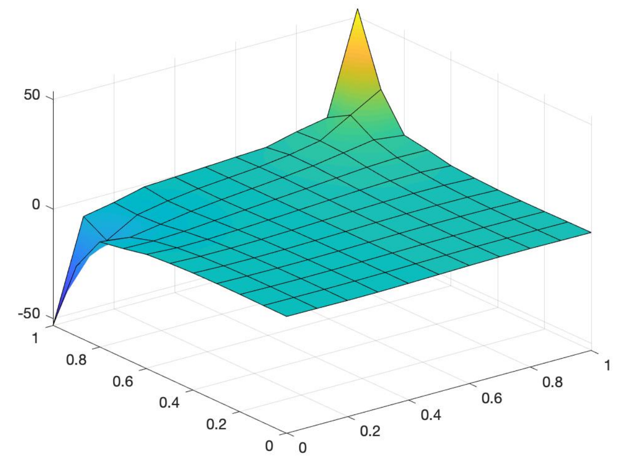

Figure 7. Pressure field for Q2P1 elements

Starting with pressure field results, as expected those discretised velocity-pressure spaces that do not satisfy the LBB condition (P1P1 and Q1Q1) present oscillations, which are stronger in the square corners. On the other hand, P2P1 and Q2Q1 discretization achieve stable results and show clearly the singularity in upper side corners.

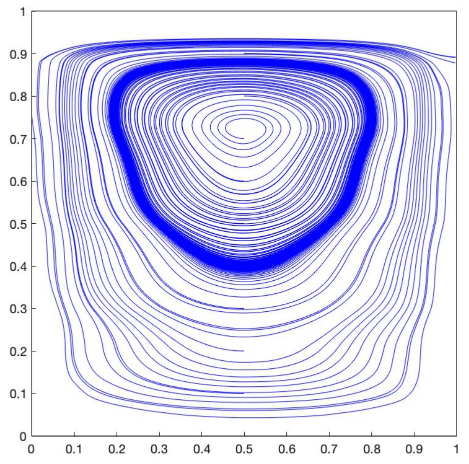

Figure 8. Streamlines Q1Q1 elements

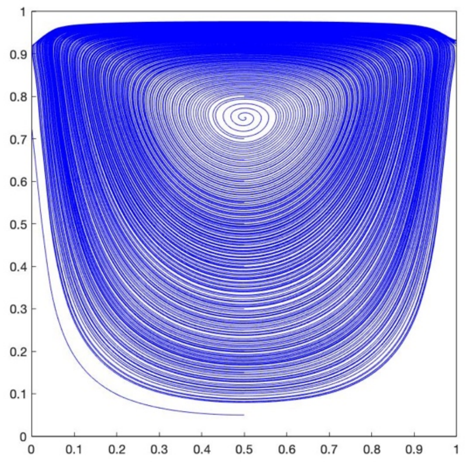

Figure 9. Streamlines Q2Q1 elements

Comparing streamlines of Q1Q1 and Q2Q1 elements, it can be seen how the first case that does not satisfy LBB condition is able to reproduce in a reasonable way streamlines and the symmetry condition with respect to the vertical centre line.

Results and conclusions – GLS formulation

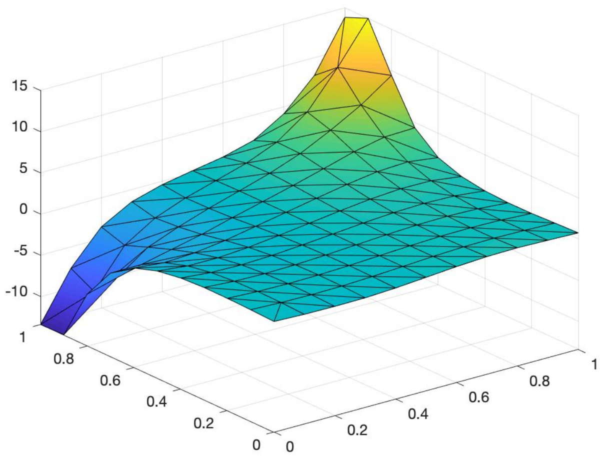

Figure 10. Pressure field for P1P1 with 10 elements

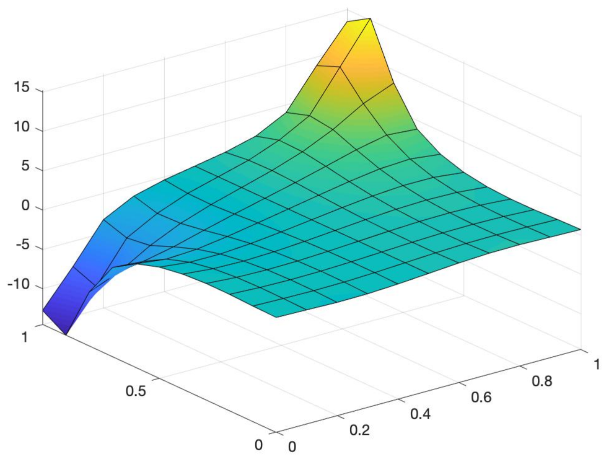

Figure 11. Pressure field for Q1Q1 with 10 elements

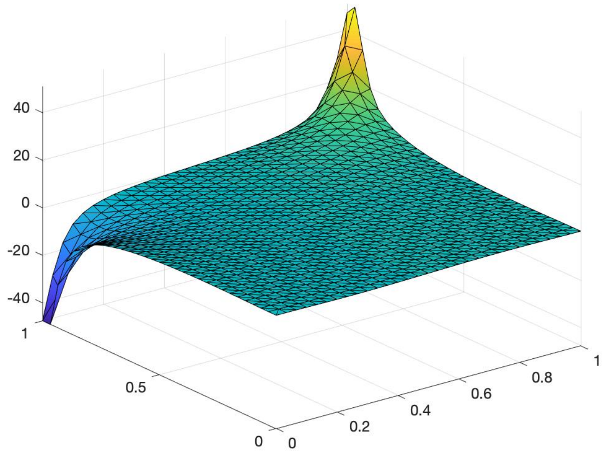

Figure 12. Pressure field for P1P1 with 30 elements

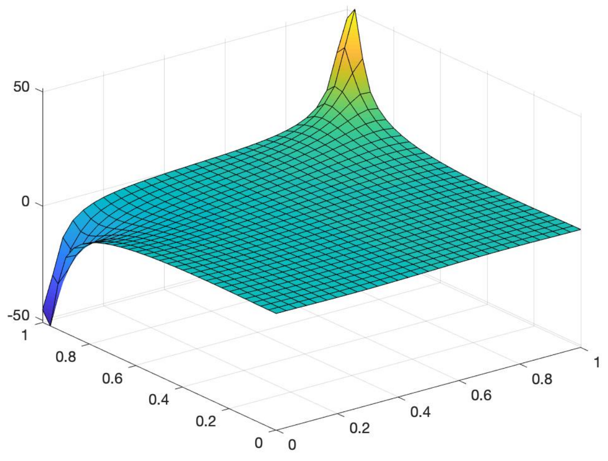

Figure 13. Pressure field for Q1Q1 with 30 elements

Taking into account the previous results, focusing in pressure the introduction of GLS formulation allows to achieve stable and better results for pressure field with those velocity - pressure pairs that don't satisfy LBB condition. At 10 element meshes (Figure 10 & 11) it can be seen that the singularity of upper corners affects more to the hole domain solution than at 10 elements Q2Q1 and P2P1 meshes (Figure 6 & 7). If the mesh is refined until 30 elements per direction, such an improvement in the results is obtained, having in mind the higher computational cost introduced non only by GLS but also by a smoother mesh.
Thus, it is prove that with help of stabilization techniques those discretization that are not stable working with classical Galerkin can be used in incompressible flow problems.

Navier – Stokes Problem

Navier – Stokes equation govern steady or transient, viscous incompressible flows. Here the steady case is going to be studied, so that the strong form is stated as

$$\begin{aligned} &-\nu\nabla^2 v + (v\cdot\nabla)v + \nabla p = b && in\,\Omega \\ &\nabla\cdot v = 0 && in\,\Omega \\ &v = v_D && on\,\partial\Omega \end{aligned}$$

Compared with the previous studied case, Stokes problem, it can be seen the presence of the non linear convective term, which is going to introduce some difficulties in the numerical simulation. Those will be overcome implementing Picard and Newton – Raphson methods, as in Burgers problem, with the main difference that now with a 2D problem, the algebra needed to achieve de Jacobian in N-R will be harder.

Discretization of Navier – Stokes equation

Applying WRM and after some algebra (integration by parts and boundary conditions), Navier – Stokes equations can be written as

$$\int_\Omega (\nabla\boldsymbol{w}):(\nu\nabla\boldsymbol{v})d\Omega + \int_\Omega \boldsymbol{w}\cdot(\boldsymbol{a}\cdot\nabla)\boldsymbol{v}d\Omega - \int_\Omega (\nabla\cdot\boldsymbol{w})pd\Omega = \int_\Omega \boldsymbol{w}\cdot\boldsymbol{f}d\Omega$$

$$\int_\Omega q\nabla\cdot\boldsymbol{v}d\Omega = 0$$

where $\boldsymbol{a}$ represents the convective velocity inside the non-linear convective term of the momentum equation. Introducing the approximation forms for velocity and pressures, as well as applying Galerkin formulation for their weighted functions $(\boldsymbol{w}, q)$ a matrix system of non-linear equations governing the discretised Navier – Stokes problem is obtained

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

K and **G** represents viscosity and discrete gradient operator respectably and **C(v)** the convection matrix.
Two different implicit methods, Picard and Newton – Raphson are going to be implement in order to solve Navier – Stokes problem dealing with the problem that works with the convective term introduces. Remark that, other modifications as GLS or SUPG stabilizations should be introduced in order to solve Galerkin classical method problem for convection dominate flows.

Picard method

Picard method can implement to solve systems as $\boldsymbol{A}(x)\boldsymbol{x} = \boldsymbol{b}(x)$. Through an iterative process, given initial value $\boldsymbol{x}^0$ the following $\boldsymbol{x}^{k+1}$ approximations are going to be solved until the solution converge.

$$\boldsymbol{A}(x^k)\boldsymbol{x}^{k+1} = \boldsymbol{b}(x)$$

Now the idea is going to solve at each iteration

$$\begin{cases} \Delta x^{k+1} = x^{k+1} - x^k \\ x^{k+1} = \boldsymbol{A}(x^k)^{-1}\boldsymbol{b}(x^k) \end{cases} \rightarrow \Delta x^{k+1} = \boldsymbol{A}(x^k)^{-1}(\boldsymbol{b}(x^k) - \boldsymbol{A}(x^k)x^k)$$

It can be applied to Navier – Stokes, obtaining the final system that will be implement in Matlab in order to obtain the desired results.

$$\begin{bmatrix} \boldsymbol{K} + \boldsymbol{C}(\boldsymbol{v}^k) & \boldsymbol{G}^T \\ \boldsymbol{G} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{v}^{k+1} \\ \boldsymbol{p}^{k+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{0} \end{bmatrix}$$

K and G matrices reaming equal than for Stokes equation and are already implement in the code, so now the focus will be in introduce the convective term inside the current code.

$$(\boldsymbol{a} \cdot \nabla)\boldsymbol{v} = \begin{bmatrix} a_x \dfrac{\partial v_x}{\partial x} + & a_y \dfrac{\partial v_x}{\partial y} \\ a_x \dfrac{\partial v_y}{\partial x} + & a_y \dfrac{\partial v_y}{\partial y} \end{bmatrix} = \begin{bmatrix} a_x & 0 & a_y & 0 \\ 0 & a_x & 0 & a_y \end{bmatrix} \cdot \begin{bmatrix} \dfrac{\partial v_x}{\partial x} \\ \dfrac{\partial v_x}{\partial x} \\ \dfrac{\partial v_x}{\partial y} \\ \dfrac{\partial v_x}{\partial y} \end{bmatrix}$$

$$w \cdot (\boldsymbol{a} \cdot \nabla)\boldsymbol{v} \rightarrow \boldsymbol{C} = \boldsymbol{N}(\boldsymbol{a} \cdot \boldsymbol{\nabla})\boldsymbol{N} \rightarrow \boldsymbol{C} = [\boldsymbol{mat\ N}]^T \begin{bmatrix} a_x & 0 & a_y & 0 \\ 0 & a_x & 0 & a_y \end{bmatrix} [\boldsymbol{grad\ N}]$$

Working with the convective term itself, it is developed to give him a shape that make things easier to pass to the discretized side. In this way, a simple scheme is obtained and introduced in Matlab code as a combination of the already defined matrices and vectors. Following the stablish scheme, two functions has been developed to compute the convective matrix. Here, the main part of element level convective matrix code is shown (Figure 1), remarking as main point that the velocity vector now must appear as an input.

```
function [Ce] = EleMatStokesConv(Xe,ngeom,nedofV,ngaus,wgp,N,Nxi,Neta,NP,v_elem)
 ...

    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);
    %Velocity
    v_ig = N_ig*v_elem;

    Ce = Ce + Ngp'*(v_ig(1)*Nx + v_ig(2)*Ny)*dvolu;
end
```

Figure 1. Main part of element convective matrix code – Picard Method

Once all the matrices are computed, just following the explained scheme of the method (already implemented) each iteration solution will be obtained.

<u>Newton – Raphson method</u>

N-R method works to solve non-linear systems as $\boldsymbol{r}(x) = \boldsymbol{0}$. Given an initial value x^0 the system is solved in an iterative process until convergence is achieved. The Jacobian plays an important role in this method, being its computation the hardest part to succeed with the method.

Given a system as $r(x) = A(x)x - b(x)$ the Jacobian is defined as $J(x) = \frac{dr(x)}{dx}$. Applying this basics concepts to discretized Navier – Stokes the way to implement the method will be obtained as follows.

$$r(x) = \begin{bmatrix}\left(K + C(v)\right)v + G^t p - f \\ Gv\end{bmatrix} \;\rightarrow\; J = \begin{bmatrix}\dfrac{dr_1}{dv} & G^t \\ G & 0\end{bmatrix}$$

Them, for eat iteration

$$J\left(x^k\right)\Delta x^{k+1} = -r(x^k)$$

Working with the J_{11} element it can be saw that all the terms, except form the last one, are already implement in Picard method.

$$\frac{dr_1}{dv} = \frac{d}{dv}\Big(\left(K + C(v)\right)v\Big) = K + C(v) + \frac{\partial C(v)}{\partial v}v$$

Inside the Jacobian, the derivation of $C(v)$ should be a combination of two components. The first one is already obtained at Picard method whereas the second one (C_2) will be obtained now in such a similar way from the initial convection term definition

$$(\boldsymbol{a}\cdot\nabla)\boldsymbol{v} = \begin{bmatrix} a_x\dfrac{\partial v_x}{\partial x} + & a_y\dfrac{\partial v_x}{\partial y} \\ a_x\dfrac{\partial v_y}{\partial x} + & a_y\dfrac{\partial v_y}{\partial y}\end{bmatrix} = \begin{bmatrix}\dfrac{\partial v_x}{\partial x} & \dfrac{\partial v_x}{\partial y} \\ \dfrac{\partial v_y}{\partial x} & \dfrac{\partial v_y}{\partial y}\end{bmatrix}\begin{bmatrix}a_x \\ a_y\end{bmatrix}$$

$$\boldsymbol{C_2} = \boldsymbol{N^T}(\boldsymbol{N}\cdot\boldsymbol{\nabla})\boldsymbol{a} \rightarrow \boldsymbol{C} = [\boldsymbol{mat\ N}]^{\boldsymbol{T}}\begin{bmatrix}\dfrac{\partial v_x}{\partial x} & \dfrac{\partial v_x}{\partial y} \\ \dfrac{\partial v_y}{\partial x} & \dfrac{\partial v_y}{\partial y}\end{bmatrix}[\boldsymbol{mat\ N}]$$

Following the stablish scheme, two functions has been developed to compute the convective matrix. Here, the main part of element level convective matrix code is shown (Figure 2) as well as the definition of the Jacobian and the increment at each step (Figure 3).

```
function [C,C2] = ConvectionMatrixNR(X,T,referenceElement,velo)
 ...

    % 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
    v_ig = N_ig*v_elem;
    M1 = nx*v_elem(:,1);
    M2 = ny*v_elem(:,1);
    M3 = nx*v_elem(:,2);
    M4 = ny*v_elem(:,2);
    M = [M1,M2;M3,M4];

    Ce = Ce + Ngp'*(v_ig(1)*Nx+v_ig(2)*Ny)*dvolu;
    Ce2 = Ce2 + (Ngp'*M*Ngp)*dvolu;

end
```

Figure 2. Main part of element convective matrix code – NR Method

```
% Computation of residual
  res = Atot*sol0 - btot;
%Jacobian
  J = [Kred+Cred+Cred2   Gred'
   Gred   zeros(nunkP)];

  % Computation of velocity and pressure increment
  solInc = -J\res;
```

Figure 3. Jacobion and step increments – NR Method

Results and conclusions

The problem will be solve using both method for a 20 elements mesh in each direction. The pair of velocity – pressure pair implemented will be Q2Q1 that satisfies de LBB condition in order to avoid problems due to convection at Galerkin formulation.

$$\begin{cases} Re = 100 \\ \nu = 1/Re \end{cases}$$

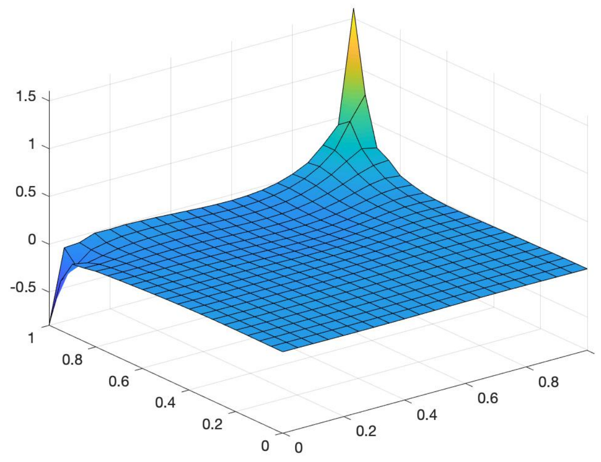

Figure 4. Pressure field – Picard method

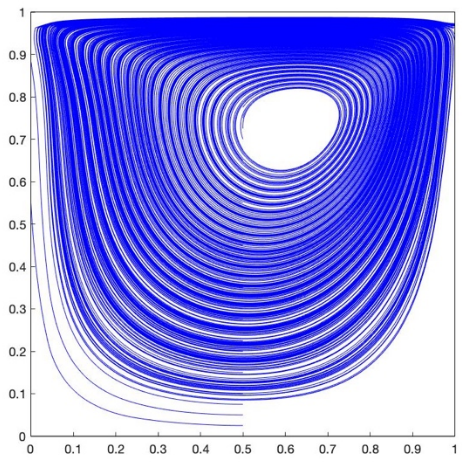

Figure 5. Streamlines – Picard method

Figures above show Picard method result, where no oscillations appear as it satisfies the LBB condition. The final result is obtained with 13 iterations and with a linear convergence plot (Figure 8), as expected.

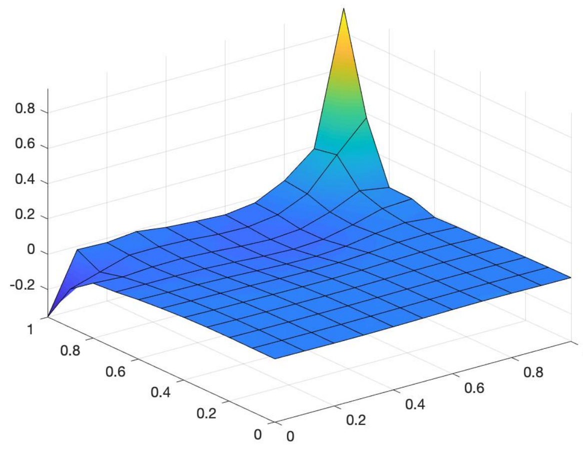

Figure 6. Pressure field – NR method

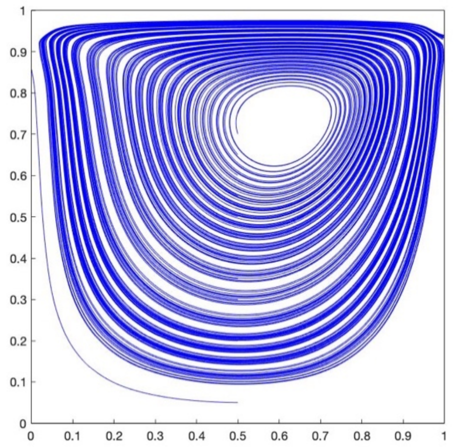

Figure 7. Streamlines – NR method

Looking now to N-R plots, as in Picard's case, no oscillations appear as it satisfies the LBB condition, just 5 iterations (Figure 9) are needed to achieve the final results. This method implies quadratic convergence, allowing the user to obtain good quality results in shorter time. Although, it is important to remark that the method itself is harder to implement and introduces higher computational cost.

Finally, convergence plots are showed to prove the different convergence behavior of the methods. Looking the plot lines make easier to understand what linear and quadratic convergence means. Check that the same error is obtained for both methods, with less iterations for NR as expected.

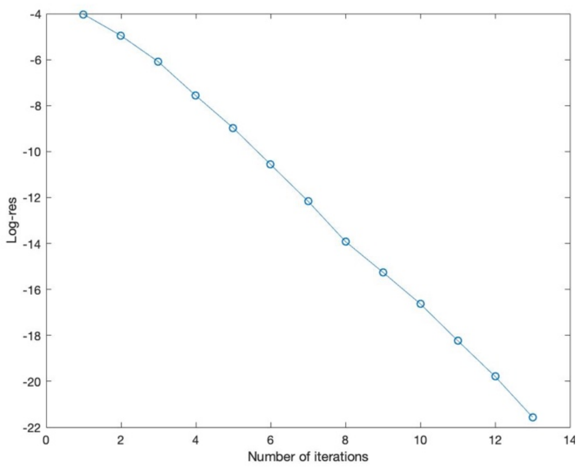

Figure 8. Picard convergence plot

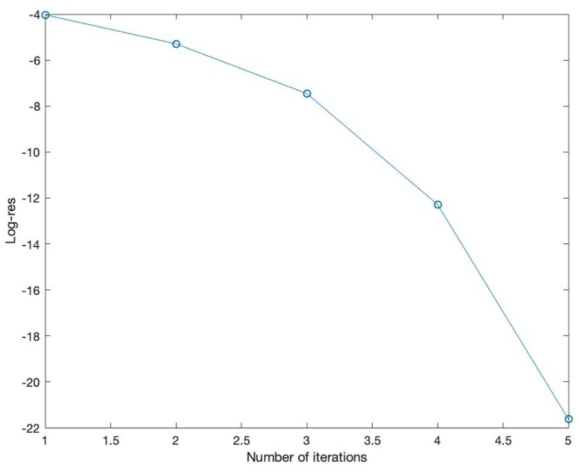

Figure 9. NR convergence plot