UPC - BARCELONA TECH
Master on Numerical Methods in Engineering
Spring 2017

Finite Elements in Fluids

Assignment # 2: VISCOUS INCOMPRESSIBLE FLOWS

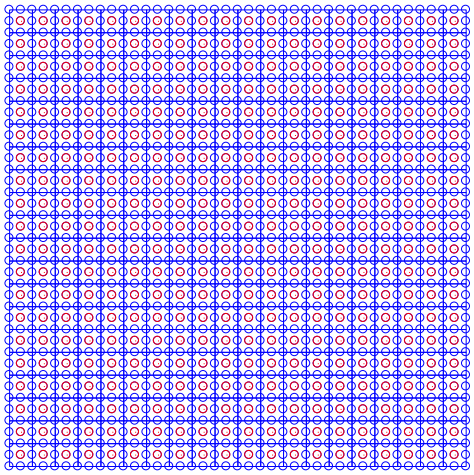
Samuel Parada Bustelo

Due date: Monday, May 29th

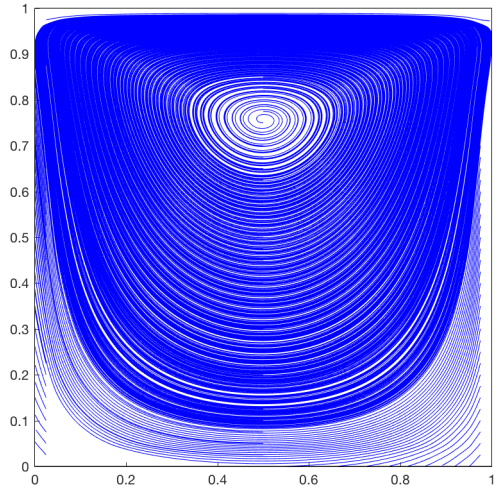
The cavity flow problem is a stander benchmark test for incompressible flows. The figure below shows a schematic representation of the problem setting. The goal of this exercise is to analyze the results obtained when adopting either the Stokes or the Navier-Stoke equations. Use the code in `HW2Files_Cavity` *to compute the finite element approximation of these problems and answer the questions below.*

(a) *Use the script* `mainStokes.m` *to compute the solution of the Stokes problem using a uniform, structured mesh of Q_2Q_0, Q_2Q_1, P_1P_1 and MINI ($P_1^+P_1$) elements, th 20 elements per side. Comment on the results.*

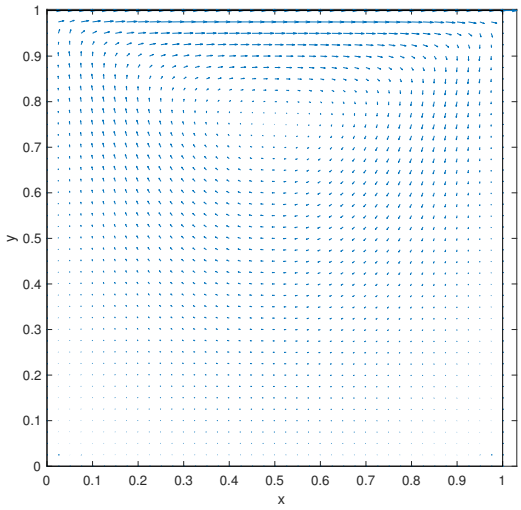
Let us present the results obtained for the cases proposed. Figure 0.1 contains the results for case of choosing Q2 elements for the velocity and Q0 elements for the pressure. As it can be seen, the main feature of this solution is than it gives a discontinuous pressure field as the pressure is contant inside the finite element. This combination does not capture quite well the solution at the upper corner.



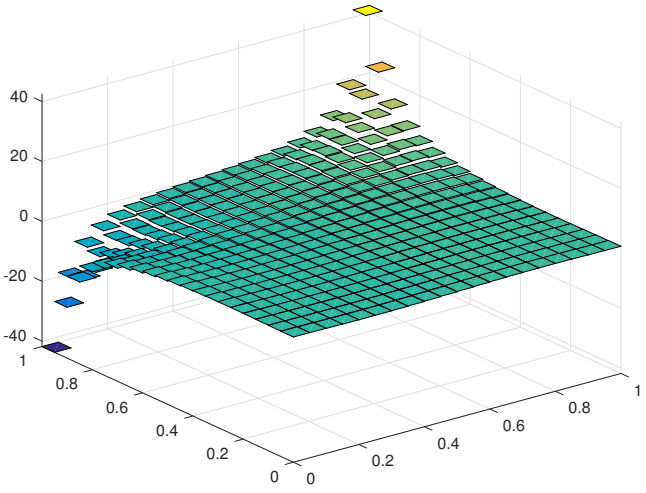
(a) Mesh of Q2Q0 elements



(b) Streamlines



(c) Velocity field

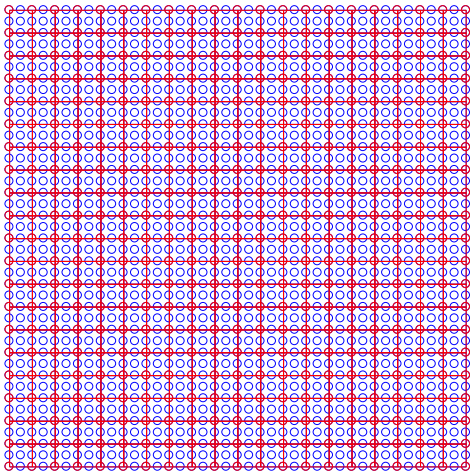


(d) Pressure field

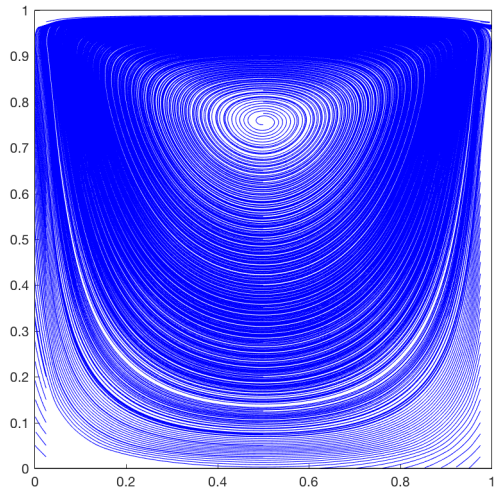
Figure 0.1: Results obtained for the case of a structured mesh of Q2 elements for velocity and Q0 elements for pressure. In (a) red nodes are for pressure and blue ones for velocity.

All in all, this element provides reliable results, as it satisfies the socalled inf-sup conditions, which guaranties the existence and uniqueness of the solution.

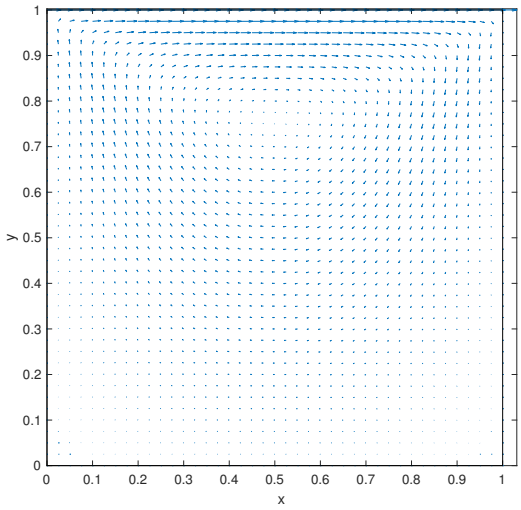
Figure 0.2 displays the results when the Q2Q1 elements are chosen, i.e. with continuous biquadratic velocity representation and continuos bilinear pressure, respectively. One can note here the symmetric character of the streamlines in Figure 0.2(b) and in the velocity field Figure 0.2(c). Recall here that the Q2Q1 element is LBB-conforming exhibiting optimal quadratic convergence. Thus, it provides accurate results. At least, more accurate than the ones presented previously for the Q2Q0 element, specially for the pressure field, which is now continuous.



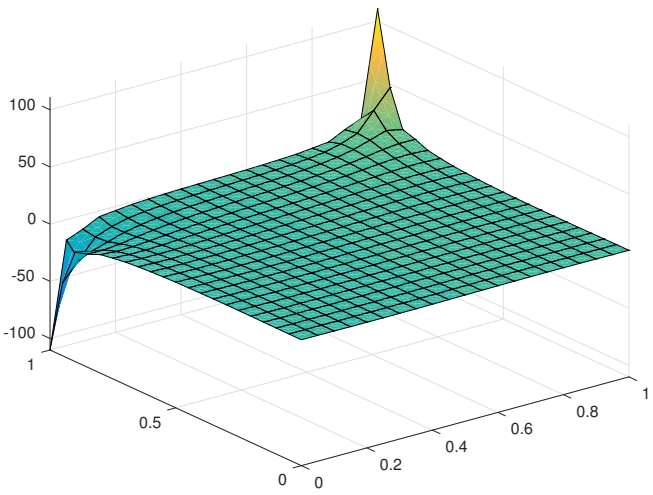
(a) Mesh of Q2Q1 elements.



(b) Streamlines



(c) Velocity field

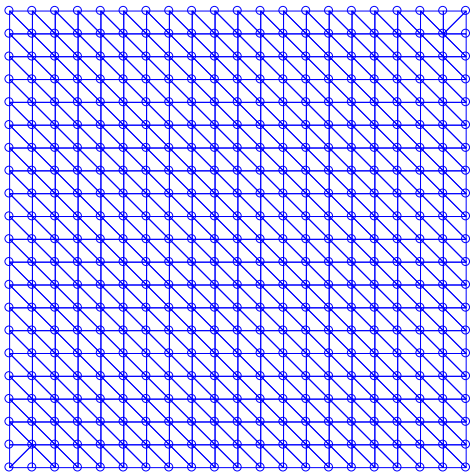


(d) Pressure field

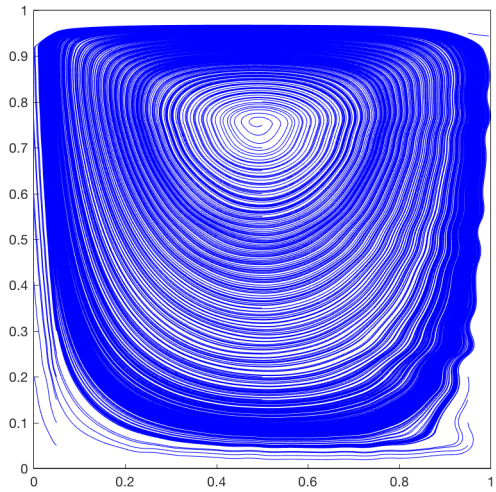
Figure 0.2: Results obtained for the Q2Q1 (Taylor-Hood) element.

Let us now present a discussion for the case of choosing a linear triangular representation for both velocity and pressure, i.e. P1P1 element. Figure 0.3 shows the results obtained after executing the code for this element. As discussed in the class, this element does not satisfy the LBB condition, which guarantees the uniqueness and existence of solution. Therefore, they present a spurious node-to-node response for the pressure field. Also, for the case of the streamlines we observe the existence of some oscillation specially on the boundaries of the mesh, Figure 0.3(b).

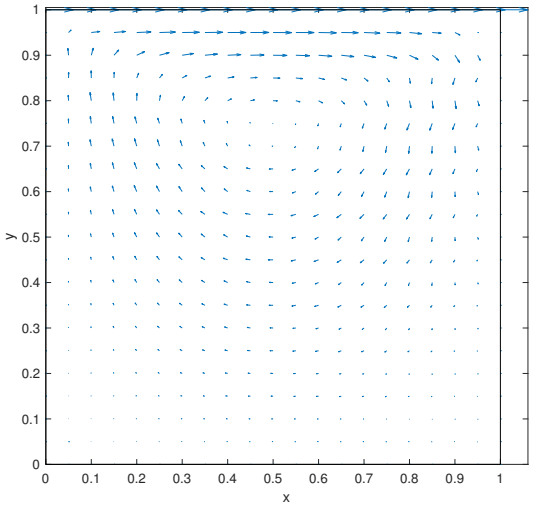
On the contrary, it is difficult to predict the spurious solution by only looking at the velocity field, Figure 0.3(c). In conclusion, this solution should be disregarded.



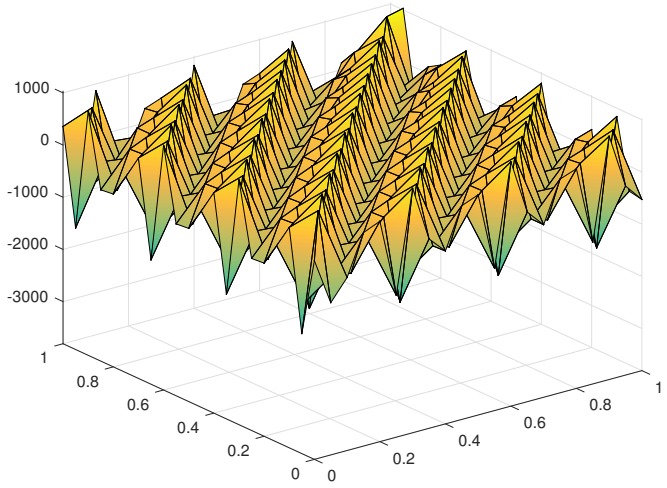
(a) Mesh of P1P1 elements



(b) Strealines



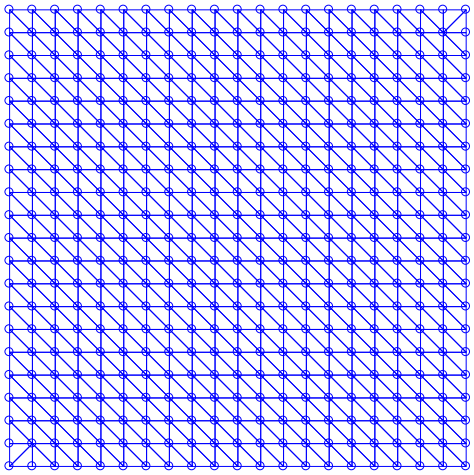
(c) Velocity field



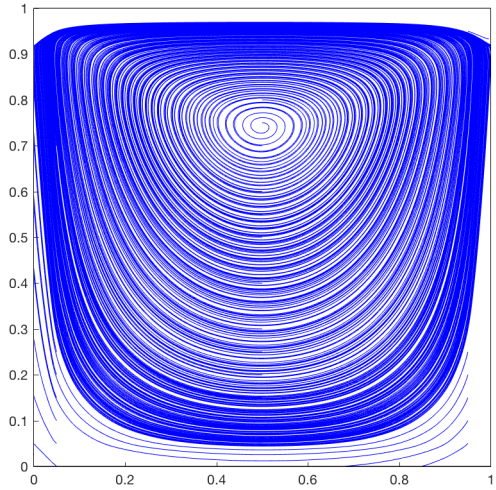
(d) Pressure field

Figure 0.3: Results obtained for the non-LBB-conforming P1P1 element.

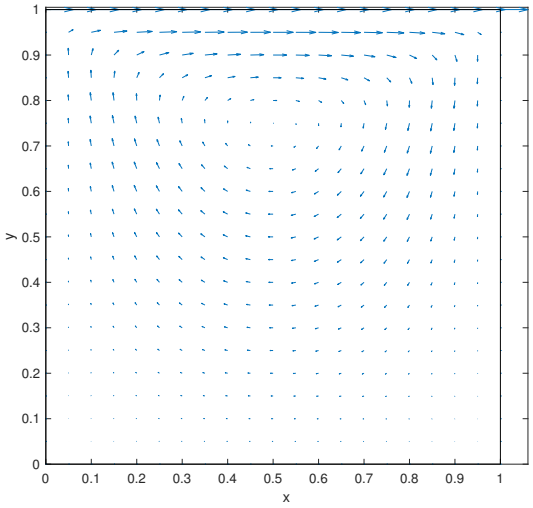
Finally, we include the results obtained for the simulation of the problem for the socalled MINI($P1^+P1$) element, Figure 0.4. The particular characteristic of this element is that it considers a piecewiese velocity field enriched with a bubble function. The pressure is continous and piecewise linear. This element is also LBB compliant and thus, as expected, we obtain reasonable results for preassure. Again note the symmetry of the streamlines with respect to the vertical axis, Figure 0.4(b).



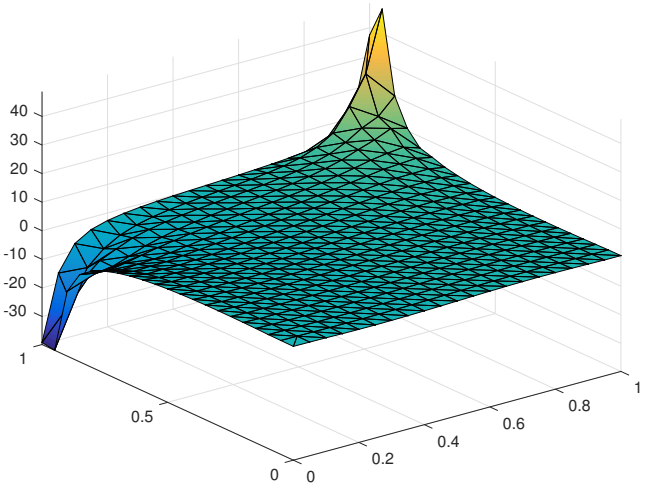
(a)



(b)



(c)



(d)

Figure 0.4: Results obtained for the MINI $P1^+P1$ element.

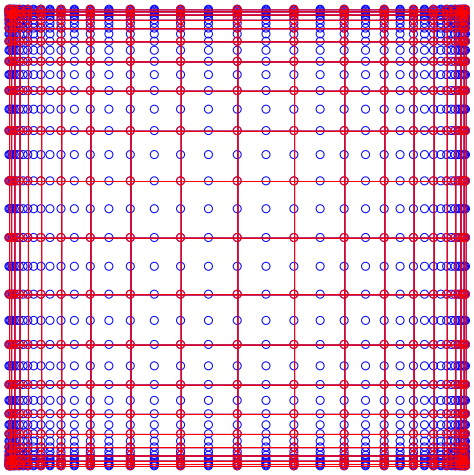
(b) *Compute the solution of the Stokes problem considering (i) a structured uniform mesh of Q_2Q_1 with 20 elements per side (ii) a structured mesh of 20x20 Q_2Q_1 elements refines near the walls. Comment on the results. Describe the main properties of the velocity and pressure fields.Are there any differences between the solutions obtained with these two meshes? Which one do you think is best? Why?*

In this section we present to analyze the effect of solving the problem when a refined mesh near the walls is considered, for Q_2Q_1 elements.

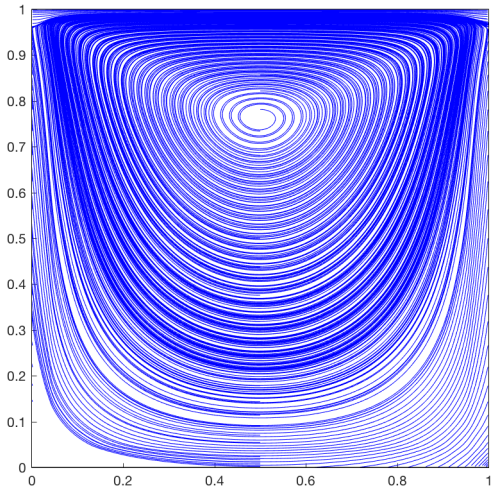
For the case of the structured uniform mesh, the results were already presentend in Figure 0.2, where it was noted the complete symmetry of the solution for streamlines and the realiable solution for the pressure field as the element is LBB-conforming. Also, recall that this element has quadratic convergence.

Figure 0.5 shows now the solution for the case of the Q2Q1 element but with a refined mesh near the walls, Figure 0.5(a). The solution for the case of the streamlines does not differ much from the case of the uniform mesh. Note the simmetry of this plot, Figure 0.5(b). On the other

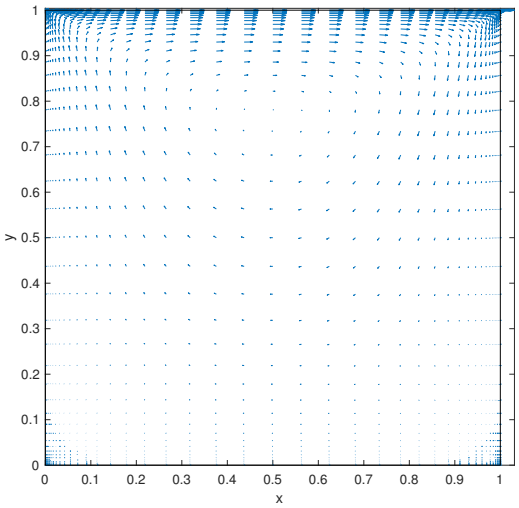
hand, for the case of the velocity field it seems like there is some difference as the velocity is more concentrated near the upper boundary. For the case of the pressure field, Figure 0.5(d), we observe that, even though both plots represent similiar shape, for the case of the refined mesh there is a smoother and thicker transition from zero to non-zero values of the pressure on the upper corner of the boundary.



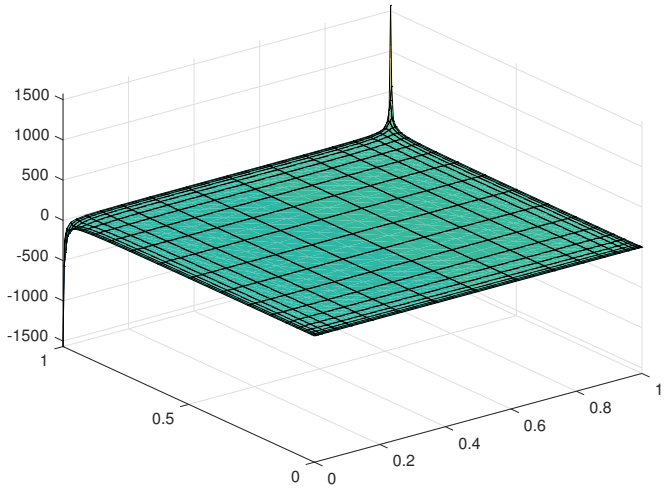
(a) Refined mesh of Q2Q1 elements.



(b) Streamlines.



(c) Velocity field.



(d) Pressure field.

Figure 0.5

For this problem, one should realize that threre is a descontinuity in the boundary conditions at the two upper cornes of the cavity. Note that velocity v_1 changes abruptly from zero value to one. Consequently, a refined mesh can be used within this zone in order to catch better the velocity change therefore providing a more accurate computation. For pressue, a smoother field is obtained.

(c) *Modify the Stokes code to solve the problem using a GLS stabilized formulation with P_1P_1 elements. Describe the formulation you are using and the choice of the stabilizaation parameter. Is the method behaving as expected?*

The stabilization of the Stokes problem pretends to enforce the positive definiteness of the final matrix of the system to be solved. This basically means to fulfill the inf-sup condition which guarantees the solution of the problem. This is done by modifying the Galerkin weak form in order to render non-zero the diagonal term resulting from the incompressiblity condition. In practice, the weak form is modified by adding the terms coming from a minization of a least squares form.

For the case of linear elements, the GLS stabilization technique does not affect the weak form of the momentun equation because terms involving second derivatives cancel. Then, the formulation can be stated as

Find $\boldsymbol{v}^h \in \mathcal{S}^h$ *and* $p^h \in \mathcal{Q}^h$*, such that, for all* $(\boldsymbol{w}^h, q^h)$ *in* $\mathcal{V}^h \times \mathcal{Q}^h$*,*

$$\begin{cases} a(\boldsymbol{w}^h, \boldsymbol{v}^h) + b(\boldsymbol{w}, p^h) = (\boldsymbol{w}^h, \boldsymbol{b}^h) + (\boldsymbol{w}^h, \boldsymbol{t}^h)_{\Gamma_N} \\ b(\boldsymbol{v}^h, q^h) - \sum_{e=1}^{n_{el}} \tau_e (\boldsymbol{\nabla} q^h, \boldsymbol{\nabla} p^h)_{\Omega^e} = -\sum_{e=1}^{n_{el}} \tau_e (\boldsymbol{\nabla} q^h, \boldsymbol{b}^h)_{\Omega^e} \end{cases}$$

But recall that for this problem does not involve neither any source term nor Neumann boundary conditions. Thus, we need to solve,

Find $\boldsymbol{v}^h \in \mathcal{S}^h$ *and* $p^h \in \mathcal{Q}^h$*, such that, for all* $(\boldsymbol{w}^h, q^h)$ *in* $\mathcal{V}^h \times \mathcal{Q}^h$*,*

$$\begin{cases} a(\boldsymbol{w}^h, \boldsymbol{v}^h) + b(\boldsymbol{w}, p^h) = 0 \\ b(\boldsymbol{v}^h, q^h) - \sum_{e=1}^{n_{el}} \tau_e (\boldsymbol{\nabla} q^h, \boldsymbol{\nabla} p^h)_{\Omega^e} = 0 \end{cases}$$

which derives into the following discrete problem,

$$\begin{bmatrix} \boldsymbol{K} & \boldsymbol{G} \\ \boldsymbol{G^T} & \boldsymbol{D} \end{bmatrix} \begin{Bmatrix} \boldsymbol{u} \\ \boldsymbol{p} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \end{Bmatrix}$$

where the Dirichlet boundary condition need to be included in order to successfully obtain the solution of the problem. Matrices $\boldsymbol{K}$ and $\boldsymbol{G}$ are already provided and matrix $\boldsymbol{D}$ can be implemented. It has the form of a typical stiffness matrix, i.e.

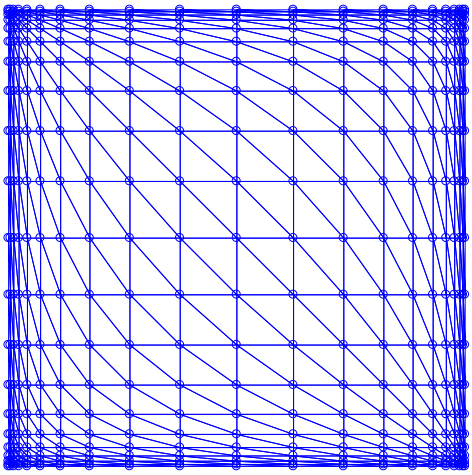
$$\boldsymbol{D} = \boldsymbol{\Lambda}^{(e)} \boldsymbol{D}^{(e)} \qquad \Rightarrow \qquad D_{ij}^{(e)} = \int_{\Omega^e} \tau_e \boldsymbol{\nabla} N_i \cdot \boldsymbol{\nabla} N_j \, d\Omega$$

The stabilization parameter is chosen as

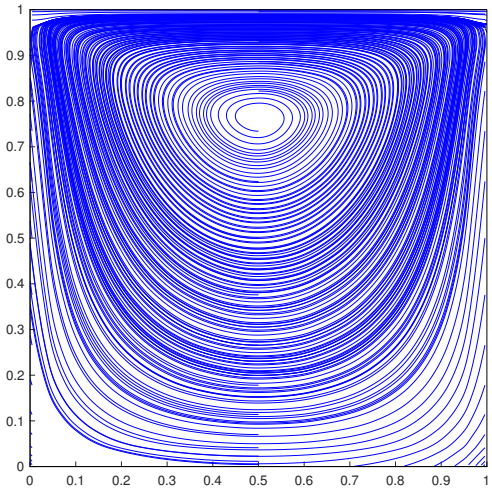
$$\tau_e = \alpha_0 \frac{h_e^2}{4\nu}$$

In this expression, h_e is an average element size and ν is the viscosity and the parameter α_0 is chosen to be equal to $1/3$, as it is proposed in [1] (chapter 6, page 288).

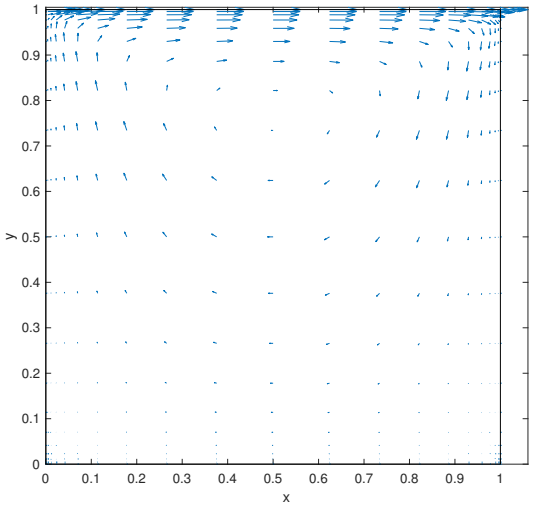
Let us now present the results obtained for the case of considering a P_1P_1 element with a GLS stabilized formulation on a mesh of 20x20 elements with refinement near the walls. Now, as one can note, the solution of this non-LBB conforming element has been stabilized and a non-spurious solution is obtained in the pressure field, in comparison with Figure 0.3(d). Also, in the case of the streamlines, in comparison with Figure 0.3(b), the image does not show some spurious solution near the boundaries and seems to be more accurate and realiable now.



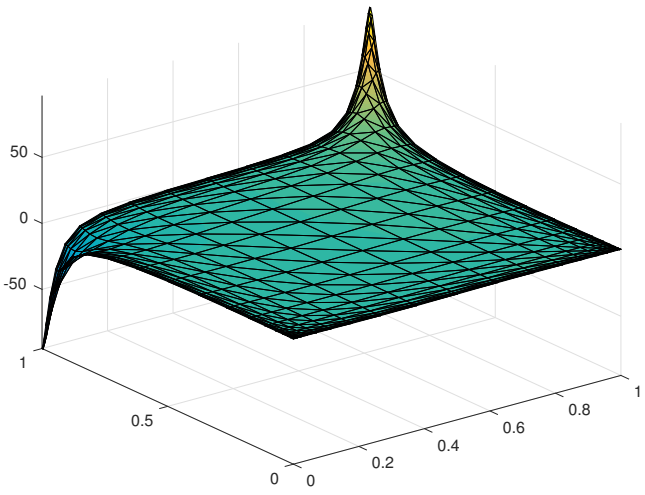
(a) Refined mesh of P_1P_1 elements.



(b) Streamlines.



(c) Velocity field.



(d) Pressure field.

Please refer to Listing 1, for a detailed *Matlab* implementation. A function for obtaining matrix $\boldsymbol{D}$ has been programmed, which is executed when it is chosen in the code.

Listing 1: GLS stabilization

```
function D = GLSMatrix(XP,TP, referenceElement,nx)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function computes the matrix arising from the discretization
% of tau*(grad q, grad p) needed for the GLS stabilization technique
%
% Inputs:
%      XP --> pressure nodes
%      TP --> connectivity of pressure nodes
%      referenceElement --> structure containing shape functions,
%      gauss points, etc...
%
%      nx --> number of elements in the discretization, needed to
%      compute tau
```

```
%
% Output:
%     D --> matrix for the GLS formulation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% Extract local variables
ngaus = referenceElement.ngaus;
wgp = referenceElement.GaussWeights;
NP = referenceElement.NP;
% I have assigned the Nxi and Neta shape functions also to the
   pressure in
% the SetReferenceElement function
NPxi = referenceElement.NPxi;
NPeta = referenceElement.NPeta;

nenP = size(TP,2); % Number of nodes per element
nPt_P = size(XP,1); % Number of nodes for pressure

% Number of degrees of freedom
nedofP = nenP;
ndofP = nPt_P;

he = sqrt(2/nx^2); % average element size
alpha = 1/3; % optimal for linear elements (Donea and Huerta, 2002)
nu=1;
tau_e = alpha*he^2/(4*nu);

D = zeros(ndofP,ndofP); % initialize matrix D

for ielem = 1:size(TP,1) % loop over elements

    TPe = TP(ielem,:); % Global number of the nodes in element ielem
    XPe = XP(TPe(1:size(TP,2)),:); % Coordinates of the nodes in
       element ielem
    TPe_dof = TPe; % Degrees of freedom in element ielem

    for ig = 1 : ngaus % loop over Gauss points

        De = zeros(nedofP,nedofP); % Initialize elemental matrix

        NP_ig = NP(ig,:);
        NPxi_ig = NPxi(ig,:);
        NPeta_ig = NPeta(ig,:);
        % Compute jacobian
        JP = [NPxi_ig(1:size(TP,2))*(XPe(:,1))  NPxi_ig(1:size(TP,2)
           )*(XPe(:,2))
              NPeta_ig(1:size(TP,2))*(XPe(:,1)) NPeta_ig(1:size(TP
                 ,2))*(XPe(:,2))];
        dvolu = wgp(ig)*det(JP);
        BP = JP\[NPxi_ig;NPeta_ig]; %matrix with derivatives

        NPx = BP(1,:);
```

```
63          NPy = BP(2,:);
64
65          De = De - tau_e*(NPx'*NPx +NPy'*NPy)*dvolu;
66      end
67
68      % Assemble the element matrix
69      D(TPe_dof,TPe_dof) = D(TPe_dof,TPe_dof) + De;
70  end
71  end
```

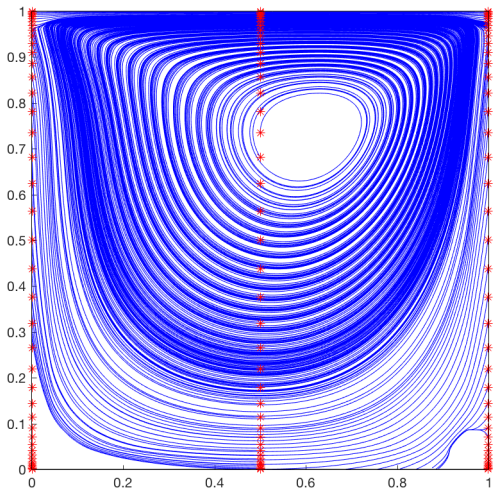
(d) *The script* ***mainNavierStokes.m*** *can be used to solve the Navier-Stokes equations with Picard method. In order to be able to use it, you must first write a Matlab function* ***ConvectionMatrix.m*** *to evaluate the matrix arising from the discretization of the convective term:*

$$c(\boldsymbol{w}, \boldsymbol{v}, \boldsymbol{v}^*) = \int_\Omega \boldsymbol{w} \cdot (\boldsymbol{v}^* \cdot \boldsymbol{\nabla})\boldsymbol{v} \; d\Omega$$

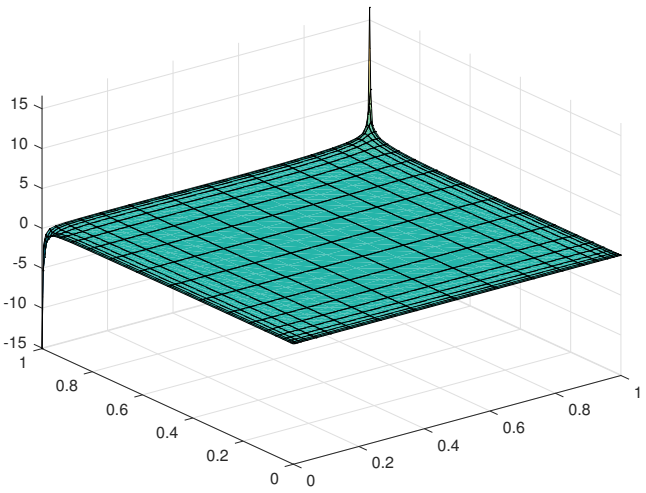
Solve the Navier-Stokes equations using a structured mesh of Q_2Q_1 elements with 20 elements per side. Consider the Reynolds numbers Re=100, 500, 1000, 2000 and comment on the results. In particular, discuss the number of iterations needed to achieve converge, the evolution of the pressure field, the position and strength of the main vortex of the velocity. Compare your results with the ones given in the literature.

In this last section we are going to present the results obtained for the Navier–Stokes for different values of the Reynold number. In order to be able to perform the calculations, the convective matrix arising from the discretization of the weak form needs to be computed first. Listing 2 offers a detailed implementation in the *Matlab* code.

For the fitst case of analysis, with $Re = 100$, 13 iterations are needed to converge.Figure 0.6(a) presents the streamlines and Figure 0.6(b) the obtained pressure field. In comparison with the Stokes problem, even though we obtain a similar pressure distribution in shape nothe that now the values of the pressure at the two upper corners are quite different. In the case of the streamlines, it seems like the main vortex has slightly moved to the right with respect ot previous solutions.



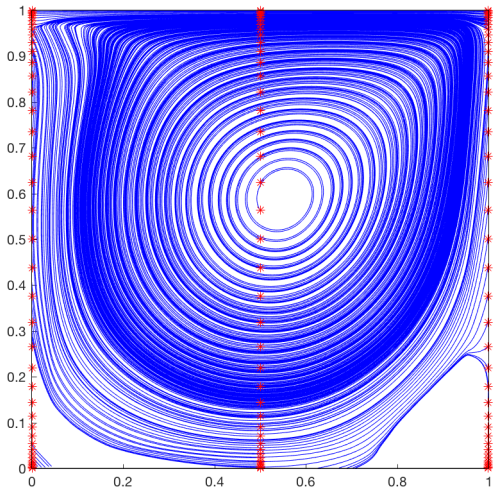
(e) Streamlines.



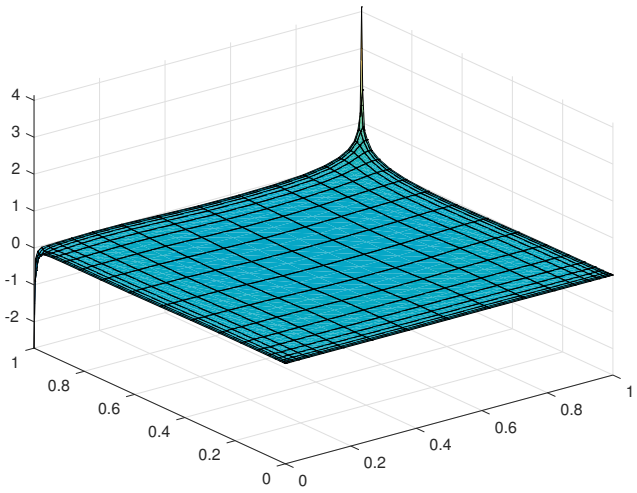
(f) Pressure field.

Figure 0.6: Results for solving the Navier–Stokes problem with Re=100. 13 iterations are needed to get a residual agreeing with the set tolerance.

In Figure 0.7 we collect the results for $Re = 500$. For this case, a total number of 29 iterations are needed to achieve a solution with the desired tolerance. The pressure field looks similar than before but one should note that the values at the two upper corners, where there is a discontinuity in the boundary condition, have changed. They are lowe now (in absolute value). Related with the streamlines, we can see that the main vortex has moved its position and is now slightly more centered than before. Further, another vortex starst to appear on the lower right corner of the domain.



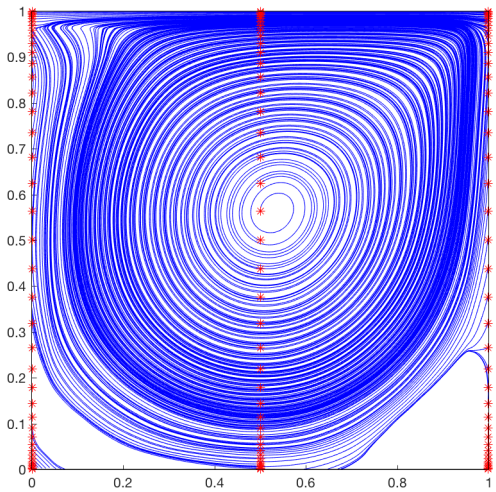
(a) Streamlines.



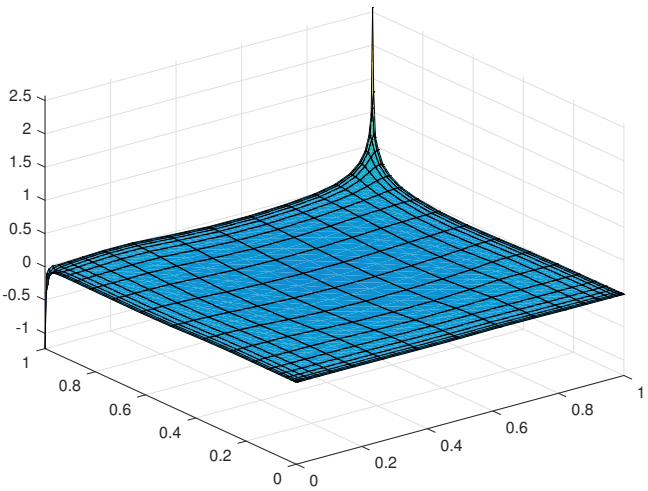
(b) Pressure field.

Figure 0.7: Results for solving the Navier–Stokes problem with Re=500. 29 iterations are needed to get a residual agreeing with the set tolerance.

Let us now present some discussion on the results for $Re = 1000$. Again, a reduction in the absolute value of the pressure is noticed, i.e. there is more suction in this case. Note that the value of the pressure in the center of the domain has decreased too.



(a) Streamlines.

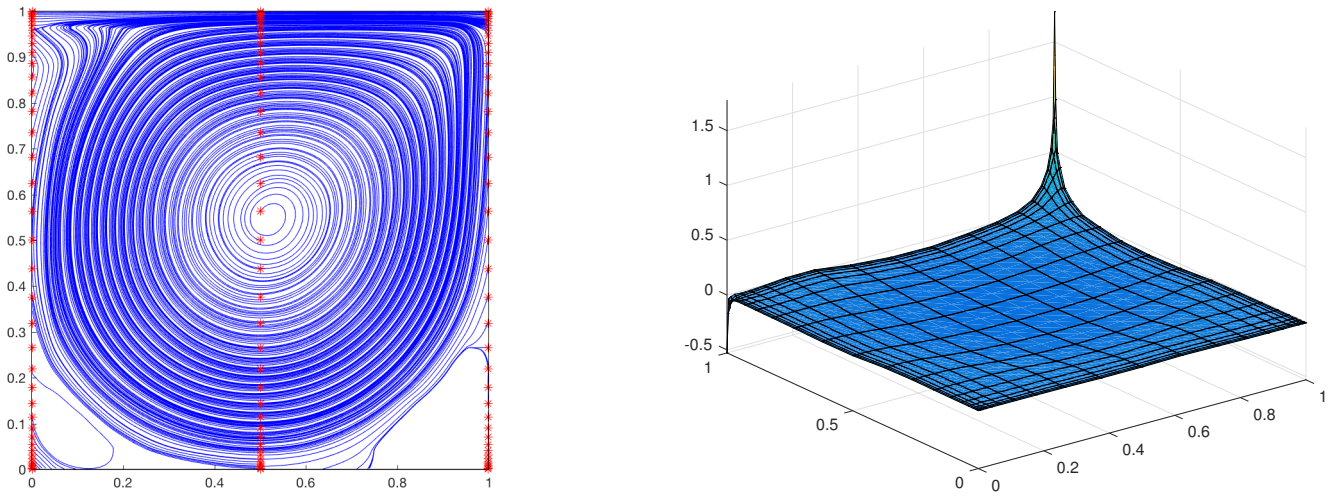


(b) Pressure field.

Figure 0.8: Results for solving the Navier–Stokes problem with Re=1000. 35 iterations are needed to get a residual agreeing with the set tolerance.

For the streamlines, the distribution is quite similar to previous case, with the main vortex centered and another one appearing on the lower right corner.

Finally, for the case of $Re = 2000$, the pressure again decreases not only on the corses but also on the center of the mesh. Morevover in the case of the streamlines, we observe the appearance of a third vortex on the lower left corner added to the two previous ones. In addition, the main vortex is even more centered now. For this case, 69 iterations are needed to achieve convergence.



(a) Streamlines.

(b) Pressure field.

Figure 0.9: Results for solving the Navier–Stokes problem with Re=2000. 69 iterations are needed to get a residual agreeing with the set tolerance.

As a conclusion, we state that increasing the Reynolds number in the computations, i.e. going to a more turbulent flows, derives into a decrease in the absolute values of the pressure field (more suction). Further, the main vortex of the velocity moves towars the center of the cavity, and a second vortex appears on the lower right corner and even a third vortex starts to come up in the lower left corner. This is because the flow becomes more turbulent. It is expected that, if we keep increasing the Re number, we would reach a point where the standard Galerkin formulation provides meaningless solutions and thus, a stabilized formulation would be requiered.

In reference [1] (chapter 6, page 312) a small table is provided with the position of the main vortex of the solution for different Reynolds number. It compares different solutions obtained in the literature for this problem like the ones from Ozawa(1975), Burggraf (1966), Tuann and Olson(1978), etc. All in all, our solution agrees with the literature as the position of the main vortex gets centered as we increase the Re number.

Listing 2: Convection matrix

```
function C = ConvectionMatrix(X,T,referenceElement,velo)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function computes the matrix arising from the discretization
% of c(w,v,v) needed for the solution of Navier-Stokes
%
% Input:
%  X --> nodal coordinates
%  T --> connectivity matrix
%  referenceElement --> data structure containing shape functions,
%  derivatives, Gauss points, etc...
%  velo --> initial velocity
```

```
12 %
13 %  Output:
14 %   C --> convection matrix
15 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
16
17 % Extract local information
18 N = referenceElement.N;
19 Nxi = referenceElement.Nxi;
20 Neta = referenceElement.Neta;
21 wgp = referenceElement.GaussWeights;
22 nnodes = referenceElement.ngeom;
23 ngaus = referenceElement.ngaus;
24
25 [numel,nen] = size(T); % Total number of elements and nodes per
      element
26 numnp = size(X,1); % Nodes in the mesh
27 ndofn = 2*nen; % Dof's per node
28 nunk  = 2*numnp; % Total number of dof's in the mesh
29
30 C = zeros(nunk, nunk);
31 for ielem = 1:numel % Loop over elements
32
33     Te = reshape([2*T(ielem,:)-1; 2*T(ielem,:)],1,ndofn);
34     Xe = X(T(ielem,1:nnodes),:);  % nodal coordinates of current
          element
35     Ve = velo(T(ielem,:),:); % velocity on nodes of current element
36     Ce = zeros(ndofn,ndofn); % initalize elemental matrix
37
38     for igaus = 1:ngaus % Loop on Gauss points
39
40         % Shape functions
41         N_igaus    = N(igaus,:);
42         Nxi_igaus  = Nxi(igaus,:);
43         Neta_igaus = Neta(igaus,:);
44
45         % Compute jacobian of the transformation
46         J = [Nxi_igaus(1:nnodes)*(Xe(1:nnodes,1))       Nxi_igaus(1:
              nnodes)*(Xe(1:nnodes,2))
47              Neta_igaus(1:nnodes)*(Xe(1:nnodes,1))      Neta_igaus
                 (1:nnodes)*(Xe(1:nnodes,2))];
48
49         dvolu=wgp(igaus)*det(J);
50         % Matrix with derivatives of shape functions
51         B = J\[Nxi_igaus;Neta_igaus];
52         % Shape functions in 2D
53         Ngp = [reshape([1;0]*N_igaus,1,ndofn); reshape([0;1]*N_igaus
              ,1,ndofn)];
54         % Derivatives
55         Nx = [reshape([1;0]*B(1,:),1,ndofn); reshape([0;1]*B(1,:),1,
              ndofn)];
56         Ny = [reshape([1;0]*B(2,:),1,ndofn); reshape([0;1]*B(2,:),1,
              ndofn)];
57         % Velocity on point igaus
```

```
58          v_igaus = N_igaus * Ve;
59          Ce = Ce + Ngp'*(v_igaus(1)*Nx+v_igaus(2)*Ny)*dvolu; %
               elemental matrix
60      end
61      C(Te,Te) =C(Te,Te) + Ce; % Assembly
62  end
63  end
```

References

[1] Donea, J., Huerta, A. *Finite Element Methods for Flow Problems*. Wiley, 2003.