Finite Element in Fluids. Assignment Part II

Adrià Galofré

May 22nd 2017

1 Problem Statement.

The cavity flow problem is a standard benchmark test for compressible 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-Stokes equations. Use the code in file in ($HW2Files_Cavity.zip$) to compute the finite elements approximation of these problems and answer the questions below

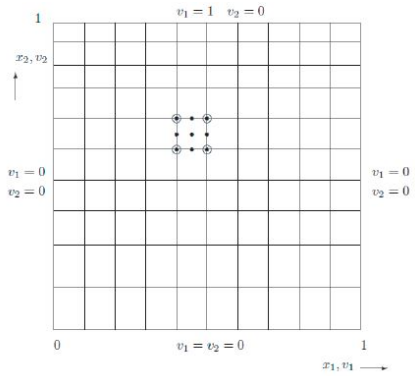
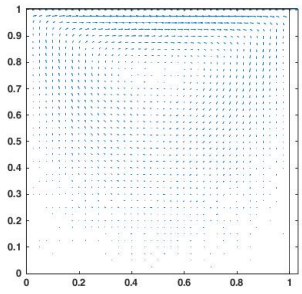


Figure 1: Problem Domain Statement

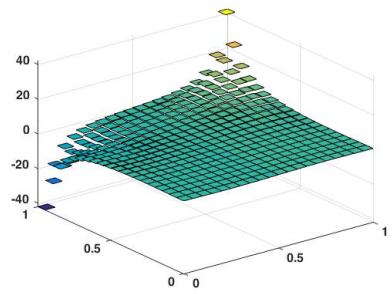
2 Questions.

2.1 Use the script *mainStokes.m* to compute the solution of the Stokes problem using a uniform, structured mesh of $Q2Q0$, Q2Q1, P1P1 and MINI (P^+1P1) elements, with 20 elements per side. Comment on the results.

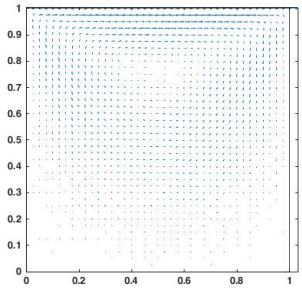
The results presented next shows the different solutions for the different pairs of velocity/pressure discretisation proposed.
The first important fact to remark is that according to the course slides not all the pairs given are admissible, so we should expect irregularities on for $Q2Q0$ and $P1P1$.
First results presented are the ones with multiquadratic approximations. It can be easily observed how the solution for the first pair, $Q2Q0$, is not continuous. As shown increasing the degree of the pressure approximation from picewise constant to multiliniear a continuous solution is obtain for the whole domain.



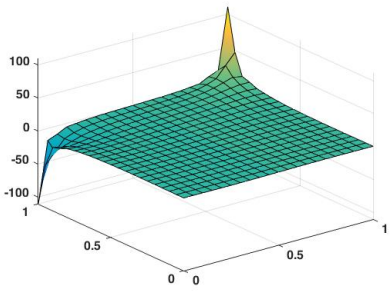
(a) $Q2Q0$ Velocity distribution.



(b) $Q2Q0$ Pressure solution.



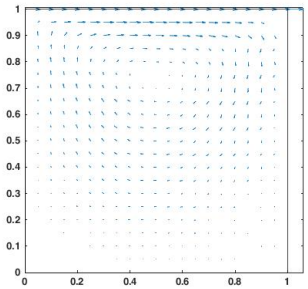
(c) $Q2Q1$ Velocity distribution.



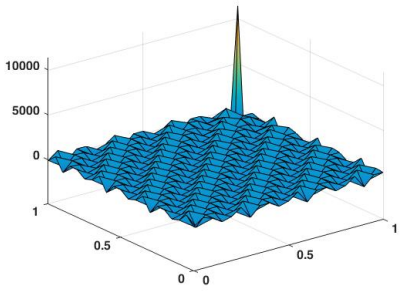
(d) $Q2Q1$ Pressure solution.

Figure 2: Results for $Q2Q0$ and $Q2Q1$ (Multiquadratic velocity).

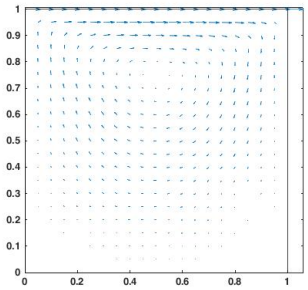
Also two pairs with linear velocity have been tested. The results obtain are shown in the image below.
The first thing noticed is that for the pair $P1P1$ some inestabilities, node-to-node oscillations, are obtained. To overcome this problem but continue using a linear velocity element, an internal enrichment with bubble functions is used. As observed the results with this enriched element provides a continuous and more stable solution.



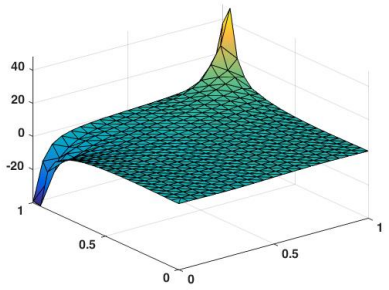
(a) P_1P_1 Velocity distribution.



(b) $P1P1$ Pressure solution.



(c) $MINI(P^+1P1)$ Velocity distribution.

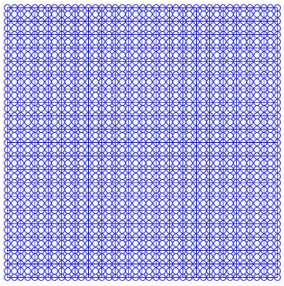


(d) $MINI(P^+1P1)$ Pressure solution.

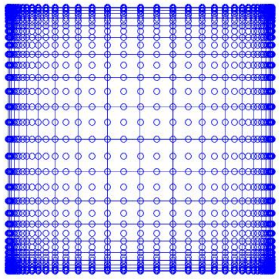
Figure 3: Results obtained for $P1P1$ and $MINI(P^+1P1)$ (Linear velocity)

2.2 Compute the solution of the Stokes problem considering (i) a structured, uniform mesh of Q2Q1 elements with 20 elements per side (ii) a structured mesh of 20x20 Q2Q1 elements refined 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 thinks the best? Why?

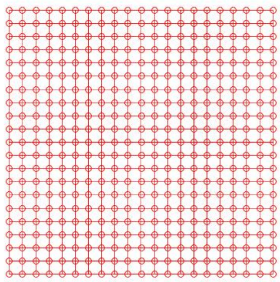
The same problem has been solved but now, as proposed we have used element Q2Q1 in other to see the influence of refining the mesh at the boundaries.
The image below shows the different meshes used:20x20 elements and structured. But one has an homogeneous distribution over the whole domain while the other one has more elements near the boundaries.



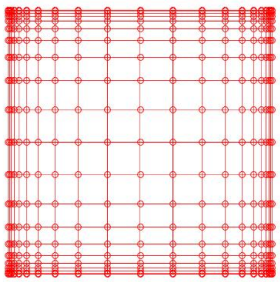
(a) Structured homogeneus velocity mesh.



(b) Structures refined over the boundaries mesh.

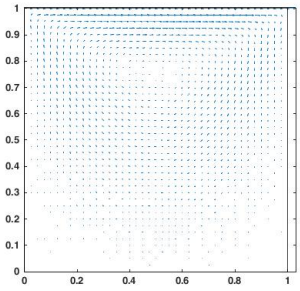


(c) Strucured homogeneus pressure mesh.

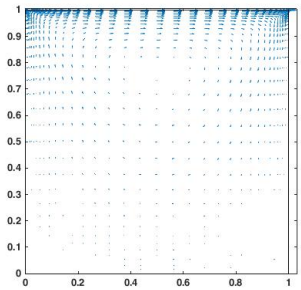


(d) Structured refined over the boundaries mesh.

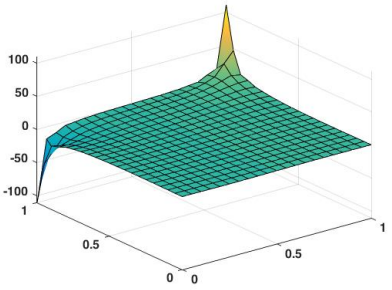
Figure 4: Different meshes used in the anaylisis.



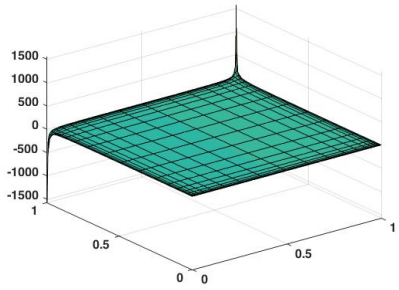
(a) Velocity distribution. Homogeneous mesh.



(b) Velocity distribution refined mesh.



(c) Pressure solution homogeneous mesh.



(d) Pressure solution refined mesh.

Figure 5: Results for the different type of meshes.

The results obtained show how this fact influences the solution obtained: For the velocity we get the same results (but distributed according to the elements distribution, of course). But is important to note how the instabilities that appeared in the pressure solution (which are in the boundaries) are now more refined so it allows us to get a more accurate solution over most of the domain.

2.3 Modify the Stokes code to solve the problem using a GLS stabilized formulation with P1P1 elements. Describe the formulation you are using and the choice of the stabilization parameter. Is the method behaving as expected?

The GLS has been implemented using the formulation described next.
The artificial diffusion added to the strong form of the problem is defined

as follows:

$$\sum_e \int_{\Omega^e} P(\omega)\tau R(u)d\Omega$$

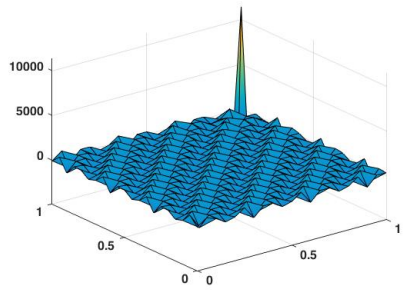
Which for the GLS stands:

$$\sum_e \int_{\Omega^e} [a \cdot \nabla\omega - \nabla \cdot (\nu\nabla\omega) + \sigma w)]\tau[(a \cdot \nabla u - \nabla(\nu\nabla u) + \sigma u) - s]d\Omega$$

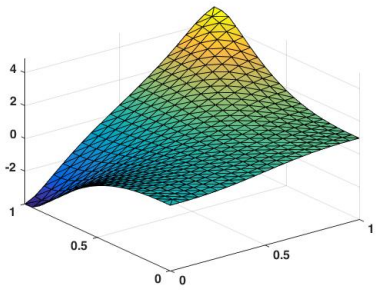
It is interesting to note that differently from other similar formulations like SUPG this artificial diffusion is symmetric and amplifies the Galerkin instabilities by . It has to be said that since we're working with linear elements and also with a no-reaction problem this terms won't affect the result (those terms can be neglected from the artificial diffusion). The stabilization parameter τ can take any value, but a good estimator of an optimal value for it was proposed by Shakib, Hughes and Johan (1991) defined by the expression:

$$\tau_P = \frac{h}{2a}\left[1 + \frac{9}{Pe^2} + \left(\frac{h\sigma}{2a}\right)^2\right]^{\frac{-1}{2}}$$

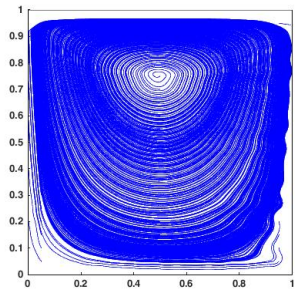
Which in this case will depend on the mesh (h), the convective parameter (a) and the Péclet number (Pe).
As observed in the results shown in the next images we solved the instability problems obtained without stabilization techniques, but the GLS introduces too crosswind diffusion.



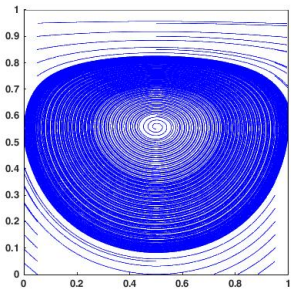
(a) Pressure distribution without GLS.



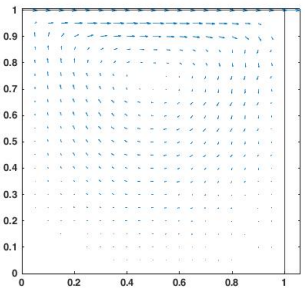
(b) Pressure distribution with GLS.



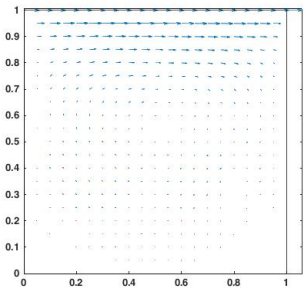
(c) Streamlines without GLS.



(d) Streamlines with GLS.



(e) Velocity distribution without GLS.



(f) Velocity distribution with GLS.

Figure 6: Results comparision with/without using GLS stabilization technique.

2.4 Solve the Navier-Stokes equations using a structured mesh of Q2Q1 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 convergence, 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 literature.

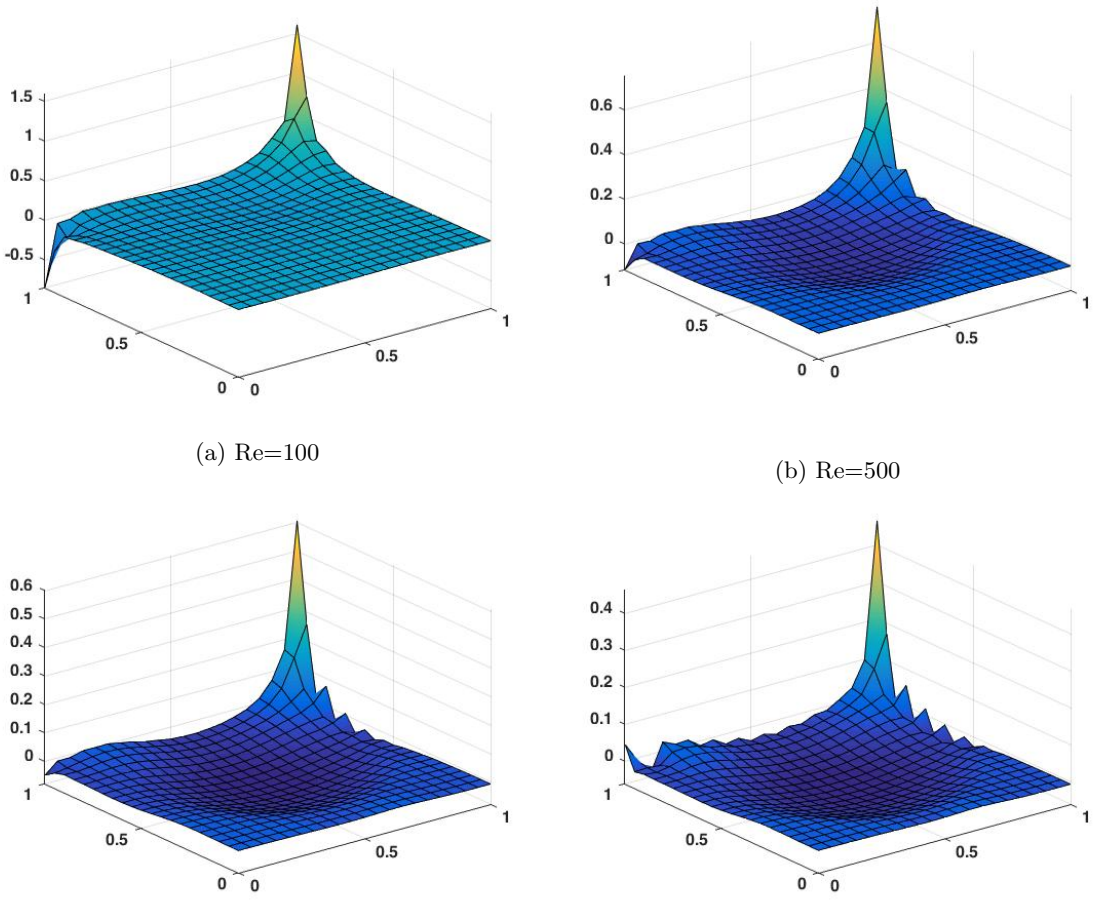
In order to solve the Navier-Stokes equations with the code provided, as explained in the assignment we first have to write the Matlab function *ConvectionMatrix.m* to evaluate the matrix arising from the discretisation of the convective term:

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

A copy of the function implemented is provided in the Annex for more details.
The discretization of the problem has been done with the mesh of 20x20. The convergence of the iterative solver for the different Reynolds numbers are:

```
Re=100     #iterations=13
Re=500     #iterations=26
Re=1000    #iterations=68
Re=2000    #iterations=99
```

As seen in the figure, the Reynolds number increases the iterations and specially the computation time required to converge increases. Despite we have reach convergence with all the Reynolds numbers proposed, it is something to take into account and think of strategies to reduce the computational cost of solving the problem with a high Reynolds number. One strategy, may be to use a coarser mesh so one will lose accuracy but will get a gain in the performance.



(a) Re=100

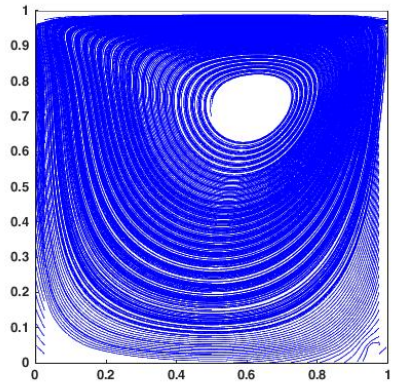
(b) Re=500

(c) Re=1000

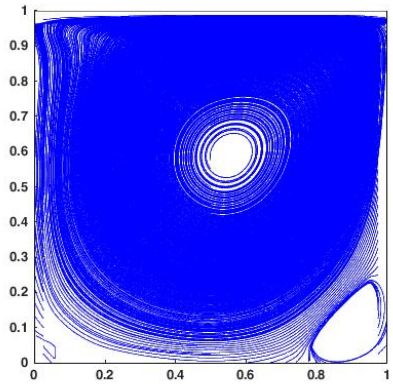
(d) Re=2000

Figure 7: Pressure solution for the different Reynold numbers.

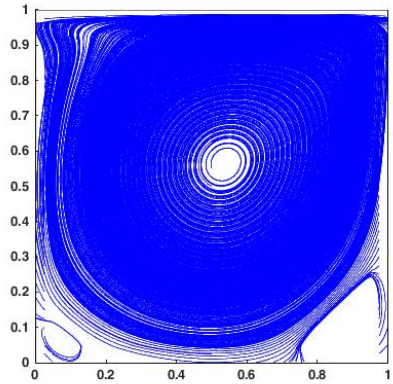
The behaviour observed in the results is due to the higher the Reynolds number is, the more imporance the intertial forces has. That is the higher pressure is applied to the fluid. In the image below, the streamlines have been also presented:



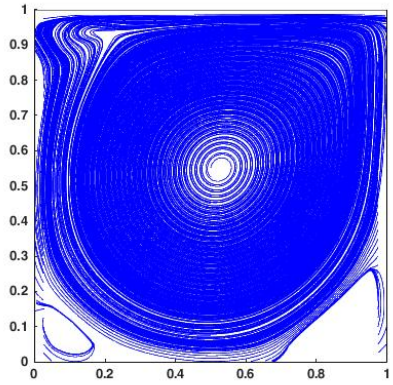
(a) Re=100



(b) Re=500



(c) Re=1000



(d) Re=2000

(e) Streamlines for different Reynolds numbers.

3 Anex. Matlab Code

3.1 GLS mainStokes.m

```
 This program solves the 2D cavity flow Stokes problem


clear; close all; clc

addpath('Func_ReferenceElement')

dom = [0,1,0,1];

% Element type and interpolation degree
% (0: quadrilaterals, 1: triangles, 11: triangles with bubble function)

% %Q2Q0
% elemV = 0; degreeV = 2;
% elemP = 0; degreeP = 0;
% %Q2Q1
% elemV = 0; degreeV = 2;
% elemP = 0; degreeP = 1;
% %P1P1
% elemV = 1; degreeV = 1;
% elemP = 1; degreeP = 1;
% %MINI(P1+,P1)
% elemV = 11; degreeV = 1;
% elemP = 1; degreeP = 1;
%P1,P1
elemV = 1; degreeV = 1;
elemP = 1; degreeP = 1;



%method=0; %without GLS stabilization
method=1;  %With GLS stabilitzation



% elemV = 1; degreeV = 2; degreeP = 1;
```

```
% elemV = 11; degreeV = 1;  degreeP = 1;
%if elemV == 11
%    elemP = 1;
%else
%    elemP = elemV;
%end
referenceElement = SetReferenceElementStokes(elemV,degreeV,elemP,degreeP);

%nx = cinput('Number of elements in each direction',20);
nx=20;
ny = nx;
h=(1/ny);
adapted = cinput('Uniform structured mesh (0)/ Structured mesh Refined at
[X,T,XP,TP] = CreateMeshes(dom,nx,ny,referenceElement,adapted);

figure; PlotMesh(T,X,elemV,'b-');
figure; PlotMesh(TP,XP,elemP,'r-');

%example variable
example.velo = [1,0];
example.a = 1;
example.nu = 1e-3;
example.sigma=0;
%example.reaction=1e-3;
example.h=h;
example.tau=(1/12)*(h^2/example.nu);
example.method=method;

% Matrices arising from the discretization
[K,G,f,Lap,fP] = StokesSystem(X,T,XP,TP,referenceElement,example);
[ndofP,ndofV] = size(G);

[dofDir,valDir,dofUnk,confined] = BC_red(X,dom,ndofV);
nunkV = length(dofUnk);
if method==0
        if confined
           nunkP = ndofP-1;
           disp(' ')
           disp('Confined flow. Pressure on lower left corner is set to zer
           G(1,:) = [];
```

```
                else
                        nunkP = ndofP;
                end
elseif method==1
        if confined
           nunkP = ndofP-1;
           disp(' ')
           disp('Confined flow. Pressure on lower left corner is set to zer
           G(1,:) = [];
           Lap(1,:) = [];
           Lap(:,1) = [];
           fP(1,:) = [];
                else
                        nunkP = ndofP;
                end
end

f = f - K(:,dofDir)*valDir;
Kred = K(dofUnk,dofUnk);
Gred = G(:,dofUnk);
fred = f(dofUnk);

if method==0
  A = [Kred    Gred';
       Gred    zeros(nunkP)];
  b = [fred; zeros(nunkP,1)];

elseif method ==1
  A = [Kred    Gred';
        Gred    Lap];
  b = [fred; fP];
end

sol = A\b;

velo = zeros(ndofV,1);
velo(dofDir) = valDir;
velo(dofUnk) = sol(1:nunkV);
velo = reshape(velo,2,[])';
pres = sol(nunkV+1:end);
```

```
if confined
    pres = [0; pres];
end

nPt = size(X,1);
figure;
quiver(X(1:nPt,1),X(1:nPt,2),velo(1:nPt,1),velo(1:nPt,2));
hold on
plot(dom([1,2,2,1,1]),dom([3,3,4,4,3]),'k')
axis equal; axis tight

PlotStreamlines(X,velo,dom);

if degreeP == 0
    PlotResults(X,T,pres,referenceElement.elemP,referenceElement.degreeP)
else
    PlotResults(XP,TP,pres,referenceElement.elemP,referenceElement.degreeP
end
```

3.2 GLS EleMatStokes$_G LS.m$

```
function [Ke,Ge,fe,Lape,fPe] = EleMatStokes_GLS(Xe,ngeom,nedofV,nedofP,ngau
% [Ke,Ge,fe] = EleMatStokes(Xe,ngeom,nedofV,nedofP,ngaus,wgp,N,Nxi,Neta,NP

Ke = zeros(nedofV,nedofV);
Ge = zeros(nedofP,nedofV);
fe = zeros(nedofV,1);
fPe = zeros(nedofP,1);
Lape=zeros(nedofP,nedofP);
% Loop on Gauss points
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))];


    dvolu = wgp(ig)*det(Jacob);
    res = Jacob\[Nxi_ig;Neta_ig];
    resP = Jacob\[NPxi_ig;NPeta_ig];

    nx = res(1,:);
    ny = res(2,:);
    nxP = resP(1,:);
    nyP = resP(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);


    %tau=0;
    %tau=2.0833e-04;
    Ke = Ke + (Nx'*Nx+Ny'*Ny)*dvolu;
    Ge = Ge - NP_ig'*dN*dvolu;
    Lape=Lape - tau*(nxP'*nxP+nyP'*nyP)*dvolu;
    x_ig = N_ig(1:ngeom)*Xe;
    f_igaus = SourceTerm(x_ig);
    fe = fe + Ngp'*f_igaus*dvolu;
    fPe = fPe - tau*(nxP'*f_igaus(1)+nyP'*f_igaus(2))*dvolu;
end
```

3.3 ConvectionMatrix.m

```
function C = ConvectionMatrix(X,T,referenceElement,velo)


elem = referenceElement.elemV;
ngaus = referenceElement.ngaus;
wgp = referenceElement.GaussWeights;
N = referenceElement.N;
```

```
Nxi = referenceElement.Nxi;
Neta = referenceElement.Neta;
ngeom = referenceElement.ngeom;

% Number of elements and number of nodes in each element
[nElem,nenV] = size(T);

% Number of nodes
nPt_V = size(X,1);
if elem == 11
    nPt_V = nPt_V + nElem;
end

% Number of degrees of freedom
nedofV = 2*nenV;
ndofV = 2*nPt_V;

C = zeros(ndofV,ndofV);

% Loop on elements
for ielem = 1:nElem
    % Global number of the nodes in element ielem
    Te = T(ielem,:);
    % Degrees of freedom in element ielem
    Te_dof = reshape([2*Te-1; 2*Te],1,nedofV);
    % Coordinates of the nodes in element ielem
    Xe = X(Te(1:ngeom),:);
    % Velocity at the nodes
    Ve = velo(Te,:);
    % Element matrix
    Ce = EleConvMatrix(Ve,Xe,ngeom,nedofV,ngaus,wgp,N,Nxi,Neta);

    C(Te_dof, Te_dof) = C(Te_dof, Te_dof) + Ce;
end
function C = ConvectionMatrix(X,T,referenceElement,velo)


elem = referenceElement.elemV;
ngaus = referenceElement.ngaus;
wgp = referenceElement.GaussWeights;
```

```
N = referenceElement.N;
Nxi = referenceElement.Nxi;
Neta = referenceElement.Neta;
ngeom = referenceElement.ngeom;

% Number of elements and number of nodes in each element
[nElem,nenV] = size(T);

% Number of nodes
nPt_V = size(X,1);
if elem == 11
    nPt_V = nPt_V + nElem;
end

% Number of degrees of freedom
nedofV = 2*nenV;
ndofV = 2*nPt_V;

C = zeros(ndofV,ndofV);

% Loop on elements
for ielem = 1:nElem
    % Global number of the nodes in element ielem
    Te = T(ielem,:);
    % Degrees of freedom in element ielem
    Te_dof = reshape([2*Te-1; 2*Te],1,nedofV);
    % Coordinates of the nodes in element ielem
    Xe = X(Te(1:ngeom),:);
    % Velocity at the nodes
    Ve = velo(Te,:);
    % Element matrix
    Ce = EleConvMatrix(Ve,Xe,ngeom,nedofV,ngaus,wgp,N,Nxi,Neta);

    C(Te_dof, Te_dof) = C(Te_dof, Te_dof) + Ce;
end
```