UPC - BARCELONA TECH
MSc Computational Mechanics
Spring 2018

Finite Elements in Fluids

Lab 6: Incompressible Navier-Stokes equations

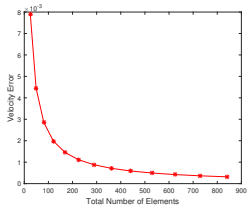
Due 1/06/2018

Prasad ADHAV

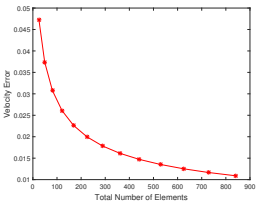
1 Stokes Problem

1.1 Computing Velocity and Pressure errors

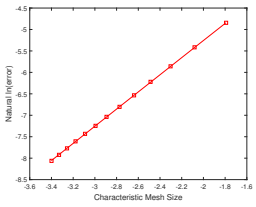
We have to compute the velocity and pressure errors, by implementing the function files in MatLAB as stated in problem statement. The convergence of Q_2Q_1 and Q_2Q_0 is to be observed.
The following figures show the convergence for the different elements, in different aspects such as number of elements and mesh size.



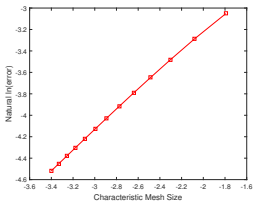
a. Vel Err- No. of Elem Q_2Q_1



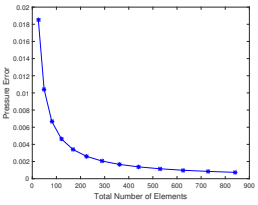
b. Vel Err Q_2Q_0



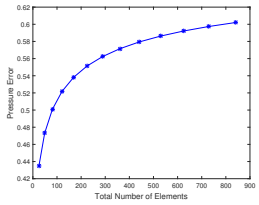
c. Vel Err- Mesh size Q_2Q_1



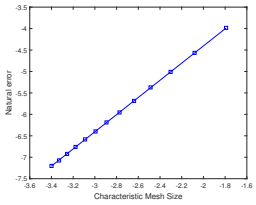
d. Vel Err Q_2Q_0



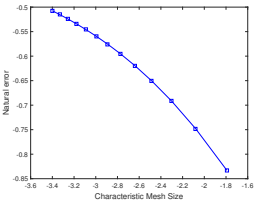
e. Prss Err-No. of Elem Q_2Q_1



f .Prss Err Q_2Q_0



g .Prss Err-Mesh size Q_2Q_1



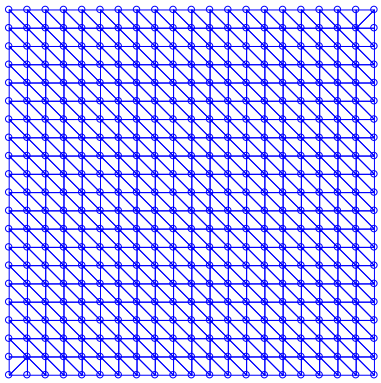
h. Prss Err Q_2Q_0

Figure 1.1 Error Graphs

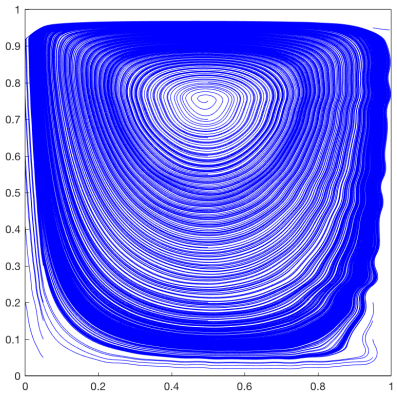
The graphs 1.1.a and 1.1.b show the variation of velocity error w.r.t number of elements. The graphs 1.1.c and 1.1.d show variation of velocity error w.r.t characteristic mesh size. The graphs 1.1.e and 1.1.f show the variation of pressure error w.r.t number of elements. The graphs 1.1.g and 1.1.h show variation of pressure error w.r.t characteristic mesh size.
As seen from the figures the errors do behave as per the theory. 1.1.a and 1.1.b it can be clearly observed that error decreases with increases in number of elements, but for element Q_2Q_1 convergence is faster than Q_2Q_0 as expected. Also, same can be observed for the velocity error w.r.t mesh size . Observe the minor difference in the slope of the graphs.
The pressure error decreases with increase in the number of elements for Q_2Q_1 element. But it is obvious that Q_2Q_0 does not behave in the expected manner, this is because of the lack of ability of Q_2Q_0 element to accurately capture the pressure behaviour. Similar characteristics can be observed from graphs 1.1.g and 1.1.h
It is clear that the Q_2Q_1 is a better of the two elements whose convergence we are observing.

1.2 Solving the problem using P_1P_1 Element

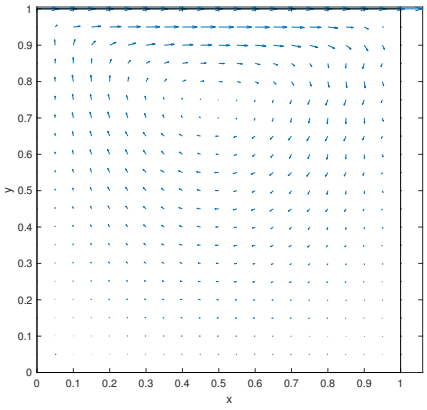
Let us now present a discussion for the case of choosing a linear triangular representation for both velocity and pressure, i.e. P1P1 element. Figure 1.2 shows the results obtained after executing the code for this element. This element does not satisfy the LBB condition, which guarantees the uniqueness 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 1.2(b).



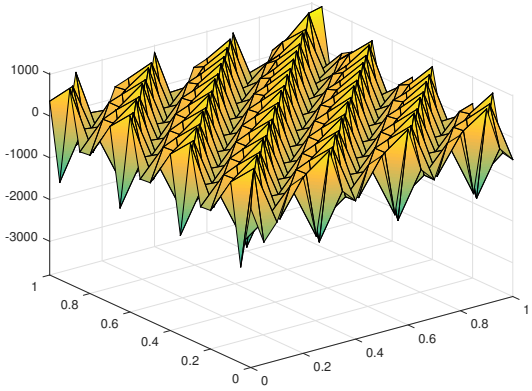
1.2(a)Refined mesh of P_1P_1



1.2(b)Streamlines



1.2(c)Velocity Field



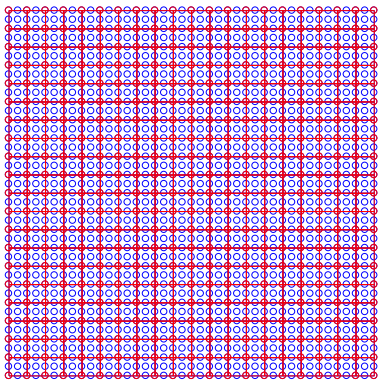
1.2(d)Pressure field

2 Cavity Flow Problem

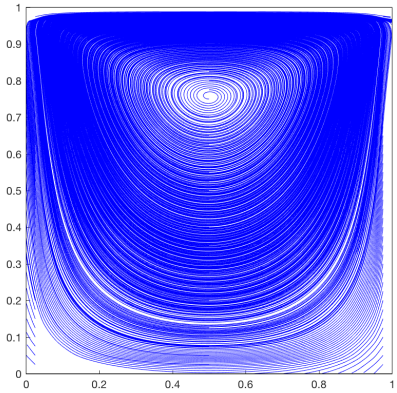
The cavity flow problem is a standard benchmark test for incompressible flows. The figure below shows a schematic representation of the problem setting. The goal of this exercise is to analyze the results obtained when adopting either the Stokes or the Navier-Stoke equations.

2.1

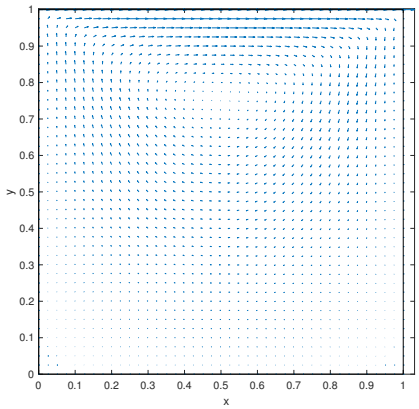
In this section the Q_2Q_1 elements with structured uniform mesh of 20×20 is considered.



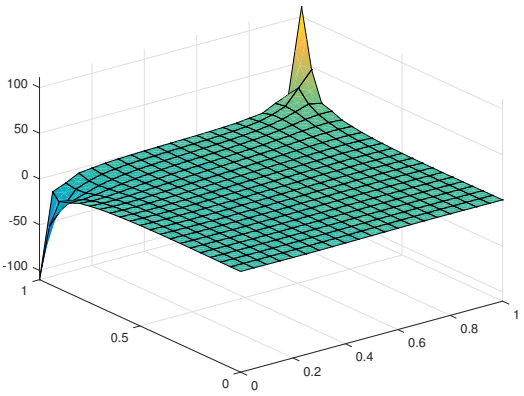
2.1(a)Refined mesh of Q_2Q_1



2.1(b)Streamlines



2.1(c)Velocity Field

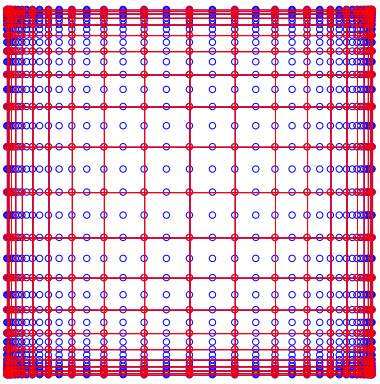


2.1(d)Pressure field

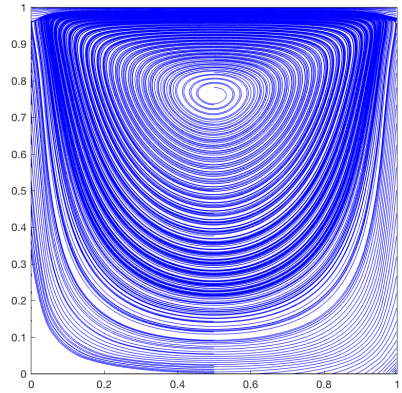
In this section we present to analyze the effect of solving the problem when a refined mesh near the walls is considered, for Q_2Q_1 elements.

For the case of the structured uniform mesh, the results were already presented in Figure 2.1, where it was noted the complete symmetry of the solution for streamlines and the reliable solution for the pressure field as the element is LBB-conforming. Also, recall that this element has quadratic convergence.

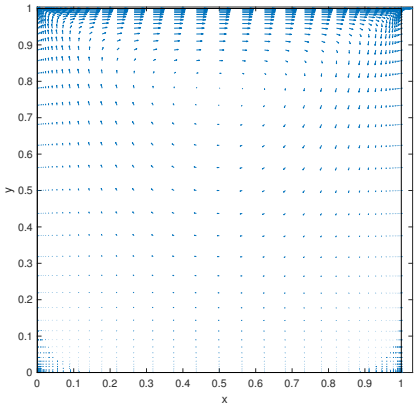
Figure shows now the solution for the case of the Q2Q1 element but with a refined mesh near the walls, Figure 2.1(e). The solution for the case of the streamlines does not differ much from the case of the uniform mesh. Note the symmetry of this plot, Figure 2.1(f). On the other hand, for the case of 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 2.1(h), we observe that, even though both plots represent similar 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.



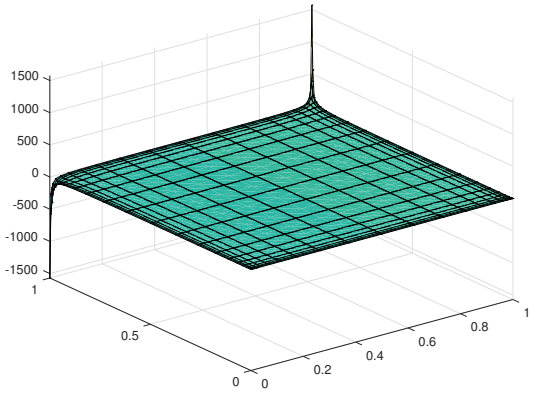
2.1(e)Refined mesh of Q_2Q_1



2.1(f)Streamlines



2.1(g)Velocity Field



2.1(h)Pressure field

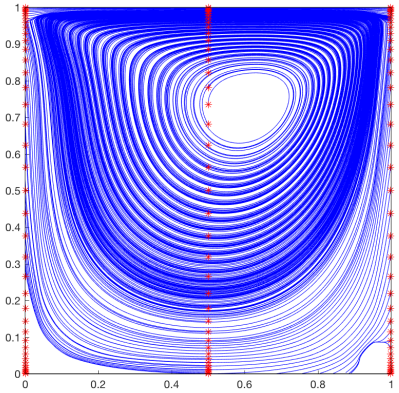
For this problem, one should realize that there is a discontinuity in the boundary conditions at the two corners of the cavity. Note that velocity v_1 changes abruptly from zero value to one. Consequently, a refined mesh can be used within this zone in order to catch better the velocity change therefore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

2.2

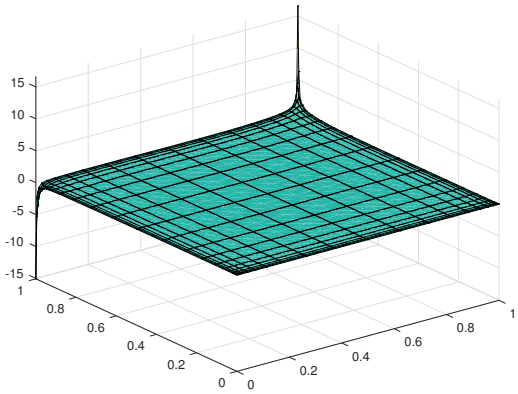
The script `mainNavierStokes.m` is used to solve the Navier-Stokes equation with Picard method. o do this a MatLAB function was first created to incorporate the matrix terms arising from discretization of the convective term as given in he problem statement. The Navier-Stokes equation is solved using a structured mesh of Q_2Q_1 elements with 20 elements per side. The Reynolds number is varied as $Re = 100, 500, 1000, 2000$.

For the first case of analysis, with Re = 100, 13 iterations are needed to converge.Figure 2.2(a) presents the streamlines and Figure 2.2(b) the obtained pressure field. In comparison with the Stokes problem, even though we obtain a similar pressure distribution in shape note that now the values of the pressure at the two corners are quite different. In the case of the streamlines, it seems like the main vortex has slightly moved to the right with respect to previous solutions.

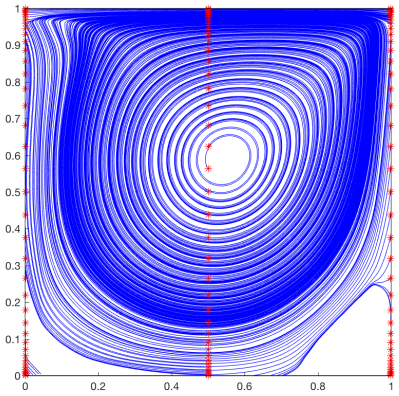
In Figure 2.2(c&d) we collect the results for Re = 500. For this case, a total number of 29 iterations are needed to achieve a solution with the desired tolerance. The pressure field looks similar than before but one should note that the values at the two upper corners, where there is a discontinuity in the boundary condition, have changed. They are low 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 starts to appear on the lower right corner of the domain.



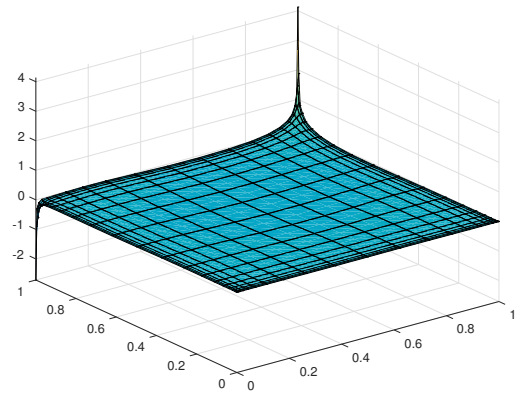
2.2(a)Streamlines Re = 100



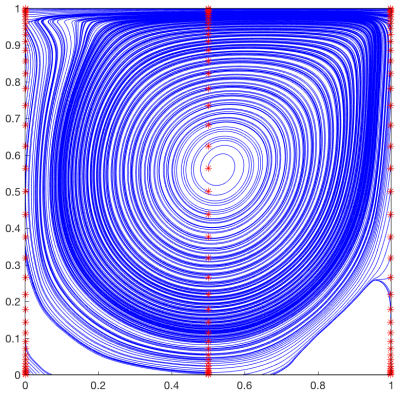
2.2(b)Pressure field Re = 100



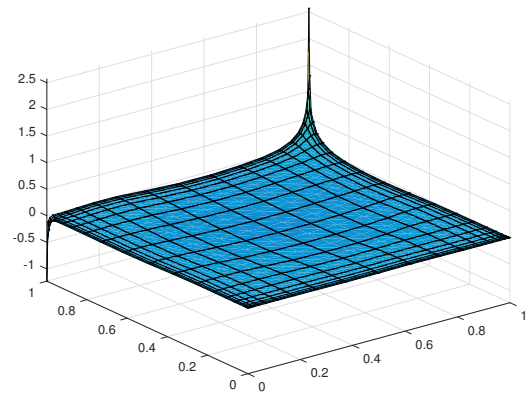
2.2(c)Streamlines Re = 500



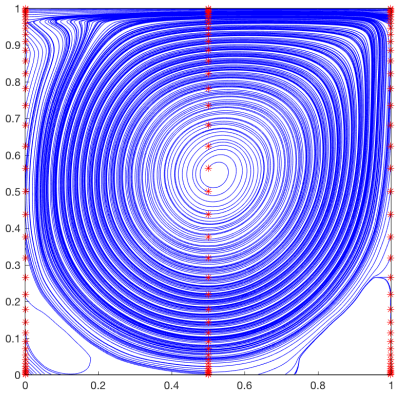
2.2(d)Pressure field Re = 500



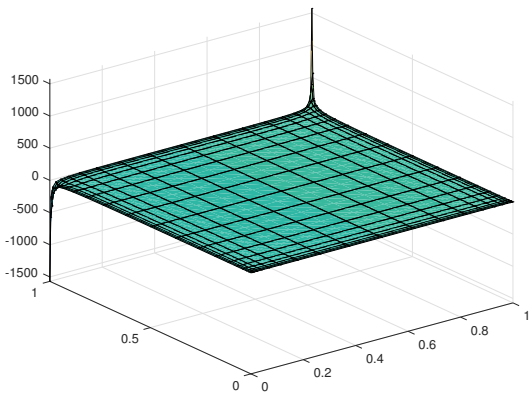
2.2(e)Streamlines Re = 1000



2.2(f)Pressure field Re = 1000



2.2(g)Streamlines Re = 2000

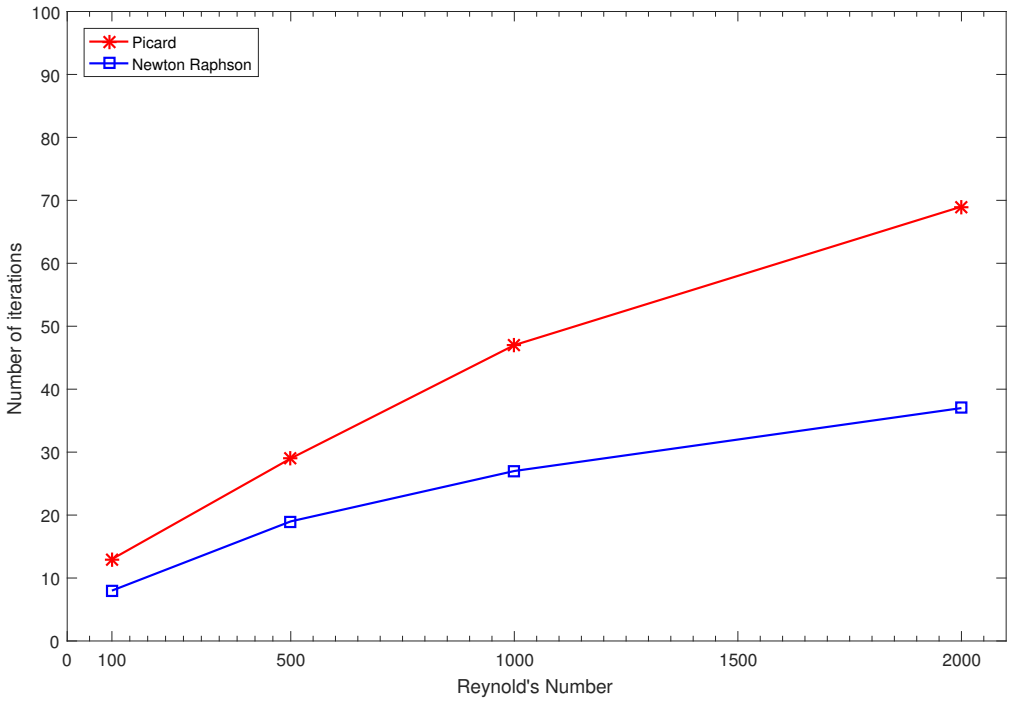


2.2(h)Pressure field Re = 2000

Now some discussion on the results for Re = 1000 presented in figure 2.2($e\&f$). 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. 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 corners but also on the center of the mesh. Moreover 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.
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 towards 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 required.

2.3

In this last section of exercise we have to write a code to solve the Cavity Flow problem with Navier-Stokes equations using Newton-Raphson method. And vary the Reynolds number as earlier, and compare the results obtained in earlier section using Picard Method.
The method is simple to code and computationally cheap, but has been known to fail or converge slowly. The Newton method is more complex and expensive (on a pre-iteration basis) than Picard.



ConvectionMatrix.m

```
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
%
%  Output:
%   C --> convection matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Extract local information
N = referenceElement.N;
Nxi = referenceElement.Nxi;
Neta = referenceElement.Neta;
wgp = referenceElement.GaussWeights;
nnodes = referenceElement.ngeom;
ngaus = referenceElement.ngaus;

[numel,nen] = size(T); % Total number of elements and nodes per element
numnp = size(X,1); % Nodes in the mesh
ndofn = 2*nen; % Dof's per node
nunk  = 2*numnp; % Total number of dof's in the mesh

C = zeros(nunk, nunk);
for ielem = 1:numel % Loop over elements

    Te = reshape([2*T(ielem,:)-1; 2*T(ielem,:)],1,ndofn);
    Xe = X(T(ielem,1:nnodes),:);  % nodal coordinates of current
       element
    Ve = velo(T(ielem,:),:); % velocity on nodes of current element
    Ce = zeros(ndofn,ndofn); % initalize elemental matrix

    for igaus = 1:ngaus % Loop on Gauss points

        % Shape functions
        N_igaus    = N(igaus,:);
        Nxi_igaus  = Nxi(igaus,:);
        Neta_igaus = Neta(igaus,:);

        % Compute jacobian of the transformation
        J = [Nxi_igaus(1:nnodes)*(Xe(1:nnodes,1))       Nxi_igaus(1:
           nnodes)*(Xe(1:nnodes,2))
             Neta_igaus(1:nnodes)*(Xe(1:nnodes,1))      Neta_igaus(1:
                nnodes)*(Xe(1:nnodes,2))];

```

```
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
```