# FEF Assignment 3

Stokes Problem

Hanna, John

## 1 Introduction

Stokes equation describes a flow where the inertial forces are very small compared to viscous forces (very low Reynolds's number), thus the inertial terms are ignored. Solving such problem using finite elements provide a difficulty. The issue is due to the fluid incompressibility which provides a constraint where the pressure is acting as a Lagrange multiplier. This is treated by having proper combination of the interpolation spaces (velocity and pressure). Instabilities might appear which is treated using stabilization techniques such as: SUPG and GLS.

## 2 Problem description and summary of the methods

The problem to be solved is a square domain 1*1 where 3 sides are fixed, while the upper one is not fixed having a prescribed velocity in the x-direction of value 1. This provides a discontinuity in the boundary conditions at the two upper corners which results in a singularity in the pressure solution as shown in the results section. The pressure is set to zero at the lower left corner.

The differential equation that is been solved is given as:

$$
\begin{gathered}
-\nu\nabla^2 v + \nabla p = b \quad in \;\; \Omega \\
\nabla . v \quad in \;\; \Omega \\
v = v_D \quad in \;\; \Gamma_D \\
-pn + \nu(n.\nabla)v = t \quad in \;\; \Gamma_N
\end{gathered}
$$

The problem is spatially discretized using the Galerkin method leading to:

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

where

$$
K = \int_\Omega [grad\ N]^T [grad\ N] d\Omega \ , \qquad G = -\int_\Omega \tilde{N}^T D d\Omega
$$

$$
F = \int_\Omega N^T f d\Omega
$$

N is the velocity shape functions, Ñ is the pressure shape functions, D is a matrix comes from the divergence operator, f represents the body forces, v and p are the velocity and pressure at nodal points.

Four type of elements are being used: bilinear in pressure and velocity quadrilateral (Q1Q1) and triangle (P1P1), biquadratic in velocity and bilinear pressure quadrilateral (Q2Q1) and triangle (P2P1). GLS stabilization technique is used for the unstable elements leading to the following system of equations for linear shape functions:

$$
\begin{bmatrix} K & G \\ G^T & L \end{bmatrix} \begin{bmatrix} v \\ p \end{bmatrix} = \begin{bmatrix} F \\ F_q \end{bmatrix}
$$

where

$$
L = \int_\Omega \tau_1 grad[\tilde{N}]^T grad[\tilde{N}] d\Omega, \qquad F_q = -\int_\Omega \tau_1 grad[\tilde{N}]^T f d\Omega
$$

and the optimal $\tau$ is

$$
tau_1 = \frac{1}{3}\frac{h^2}{4\nu}
$$

## 3 Results and Discussion

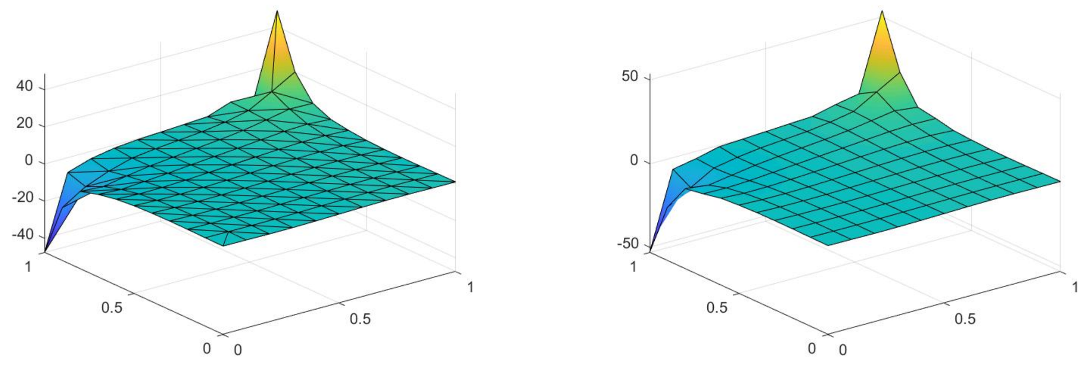

Figure 1: stable Pressure solution for P2P1 and Q2Q1

The above graphs show the pressure results for P2P1 and Q2Q1 elements. The solution is smooth and no instabilities are present.

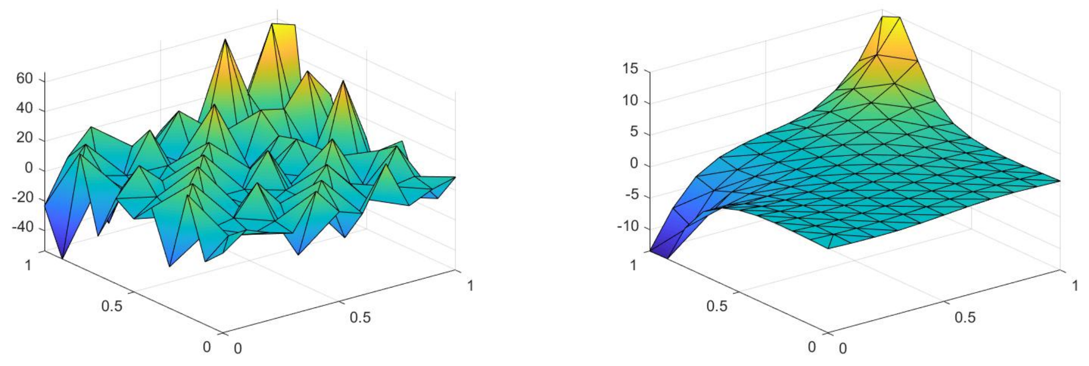

Figure 2: Unstable and GLS stabilized pressure solution for P1P1

The above graphs show the pressure results for P1P1 and the same element with GLS stabilization applied. Without stabilization, instabilities are clear using linear interpolation for both velocity and pressure.

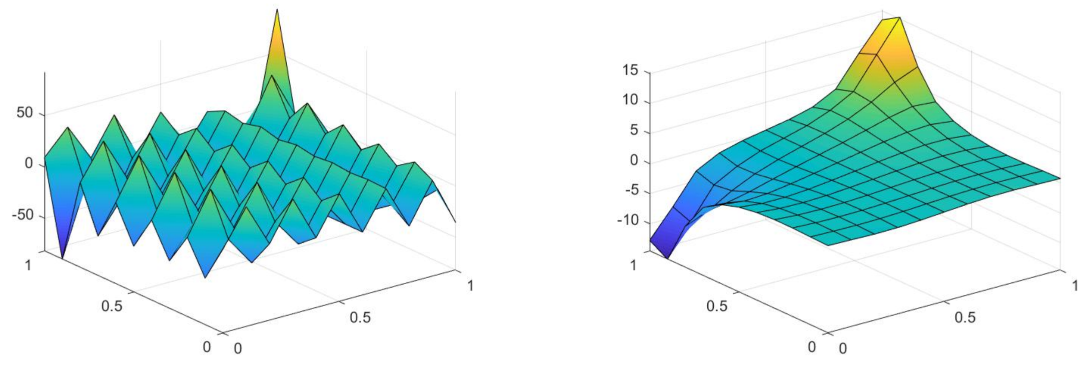

Figure 3: Unstable and GLS stabilized pressure solution for Q1Q1

The above graphs show the pressure results for Q1Q1 and the same element with GLS stabilization applied. Same instabilities are clear using linear interpolation for both velocity and pressure.

## 4 Conclusion

To conclude, the incompressability condition in Stokes problem can provide instabilities in the solution if inappropriate combination of velocity and pressure interpolation is chosen such as: linear interpolation in both spaces. To solve this issue, SUPG or GLS stabilization techniques can be applied to the system of equations to provide the required stability.

## 5 Appendix(Different parts of GLS implementation)

### 5.1 Matricies Calculation

```
Ke = Ke + (Nx'*Nx+Ny'*Ny)*dvolu;
Ge = Ge - NP_ig'*dN*dvolu;
Le = Le - tau*(nx'*nx+ny'*ny)*dvolu;  % added term
x_ig = N_ig(1:ngeom)*Xe;

f_igaus = SourceTerm(x_ig);

fe = fe + Ngp'*f_igaus*dvolu;
feq = feq + [nx; ny]'*f_igaus*dvolu;  % added term
```

### 5.2 System Solving

```
% Total system of equations
if confined
    nunkP = ndofP-1;
    disp(' ')
    disp('Confined flow. Pressure on lower left corner is set to zero');
    G(1,:) = [];
    L(1,:) = [];          % modified
    L(:,1) = [];          % modified
else
    nunkP = ndofP;
end

A = [Kred    Gred';
    Gred    Lred];                          %modified
b = [fred; fqred];

sol = A\b;
```

# Navier-Stokes Problem

## 1 Introduction

Steady Navier-Stokes equation describes general viscous steady Newtonian flows. Solving such problem using finite elements provide several difficulties. The first one is fluid incompressibility which provides a constraint where the pressure is acting as a Lagrange multiplier. Another problem is the nonlinear convective term that provides instabilities. Moreover, since the convective term is nonlinear, iterative methods are used to solve the problem such as: Picard and Newton-Raphson methods. The last issue is treated in this report.

## 2 Problem description and summary of the methods

The differential equation that is been solved is given as:

$$
\begin{gathered}
-\nu\nabla^2 v + (v.\nabla)v + \nabla p = b \quad in \;\; \Omega \\
\nabla .v \quad in \;\; \Omega \\
v = v_D \quad in \;\; \Gamma_D \\
-pn + \nu(n.\nabla)v = t \quad in \;\; \Gamma_N
\end{gathered}
$$

The problem is spatially discretized using the Galerkin method leading to:

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

where

$$
Cv = \int_\Omega w(v.\nabla)v \; d\Omega
$$

As shown the convective term provides a nonlinear term in v. Two iterative methods are used. The first one is Picard method explained as follows. First, the system of equations to be solved can be written as:

$$
A(v)v = b
$$

An initial guess is given to the solution vector as $v^0$, then the next iteration is calculated as follows.

$$
v^{k+1} = A^{-1}(x^k)b(x^k)
$$

This is done till $\Delta v = v^{k+1} - v^k$ is very small according to a given tolerance.

The second method is Newton Raphson. Having an initial guess, the residual vector is calculated as follows:

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

Then the Jacobin Matrix is calculated as:

$$
J = \begin{bmatrix} \frac{\partial r_1}{\partial v} & G^T \\ & \\ G & 0 \end{bmatrix}
$$

The first term in the Jacobin matrix is the source of difficulty since it involves the nonlinear function in v. After deriving the term, the system of equations to be solved will be:

$$
\begin{bmatrix} K + C(v^k) + C_2(v^k) & G \\ G^T & 0 \end{bmatrix} \begin{bmatrix} v \\ p \end{bmatrix} = \begin{bmatrix} F + Fc(v^k) \\ 0 \end{bmatrix}
$$

$$
C_2 v^{k+1} = \int_\Omega w \nabla v^k v^{k+1} \; d\Omega, \quad F_c = \int_\Omega w \nabla v^k v^k \; d\Omega
$$

## 3 Results and Discussion

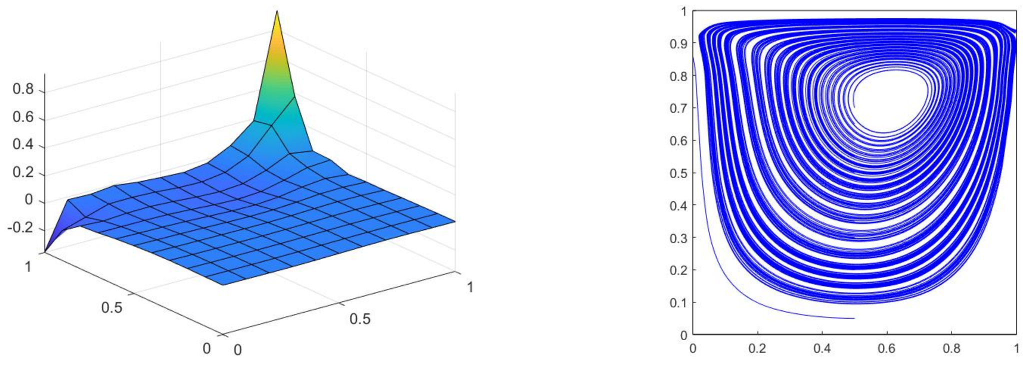

Figure 1: Pressure and streamlines using Picard method

The solution using Picard method above converged using 13 iterations. It should be noted that the Navier-Stokes solution didn't give us a symmetric solution as in Stokes problem.

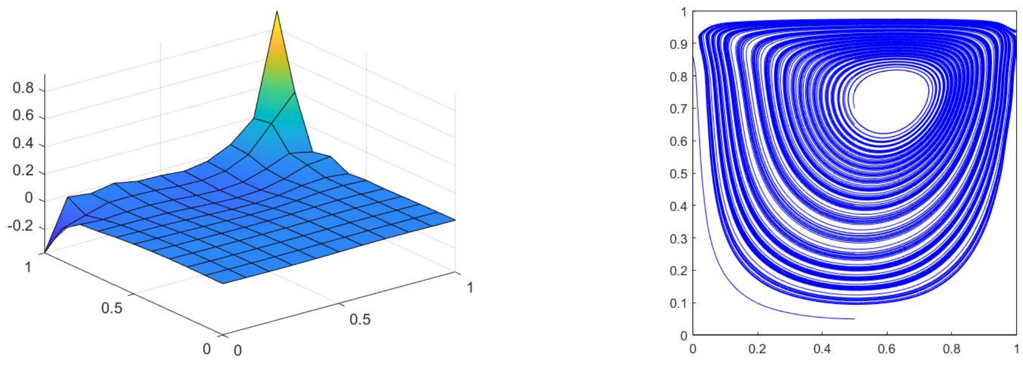

Figure 2: Pressure and streamlines using NR method

Similar results were obtained using Newton-Raphson method as shown above.

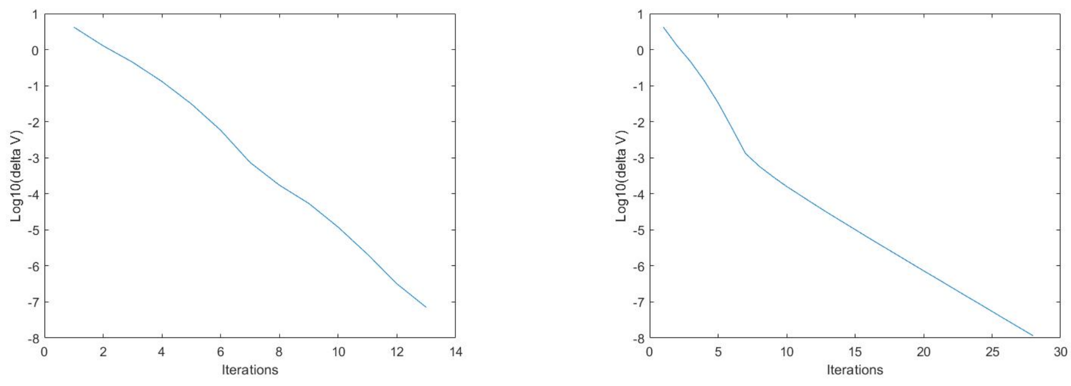

Figure 3: Convergence of Picard and NR methods

As shown above, Picard method took about 13 iterations and it's linear which agrees with the theory. While Newton-Raphson took 28 to converge and the behaviour was not quadratic which doesn't agree with the theory. Therefore, an error in the implementation might have occurred in solving the problem.

## 4 Conclusion

To conclude, solving Navier-Stokes equations using finite elements will lead to nonlinear system of equations. Therefore, iterative methods should be used such as: Picard and NR methods. Newton Raphson provides quadratic convergence but it's harder to implement than Picard method.

## 5 Appendix(MATLAB codes)

### 5.1 Matrices calculation

```
vel_ig = N_ig*vel;

Ce = Ce + Ngp'*( vel_ig(1)*Nx + vel_ig(2)*Ny)*dvolu;
Ce2 = Ce2 + Ngp'*( vel_ig(1)*Nx + vel_ig(2)*Ny)*dvolu....
        + Ngp'*(Nx*velfx + Ny*velfy)*Ngp*dvolu;
fce = fce + Ngp'*(Nx*velffx.^2 + Ny*velffy.^2)*dvolu;

%% velfx and velfy are differnt shapes of the known velocity vector in x and y
```

### 5.2 Newton-Raphson

```
Cred = C(dofUnk,dofUnk);
Cred2 = C2(dofUnk,dofUnk);
fcred = fc(dofUnk);

r=[(Kred+Cred)*sol0(1:nunkV)+Gred'*sol0(nunkV+1:end)-fred; Gred*sol0(1:nunkV)];
J=[Kred+Cred2,Gred';Gred,zeros(nunkP)];

Atot = A;
Atot(1:nunkV,1:nunkV) = A(1:nunkV,1:nunkV) + Cred;
btot = [fred - C(dofUnk,dofDir)*valDir; zeros(nunkP,1)];

% Computation of velocity and pressure increment
solInc = -inv(J)*r;
```