FINITE ELEMENTS IN FLUIDS

Master of Science in Computational Mechanics/Numerical Methods

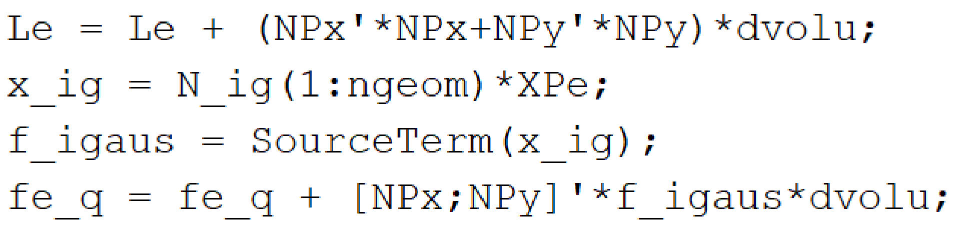
Spring Semester 2019

Rafel Perelló i Ribas

Laboratory: 3rd session

Stokes flow

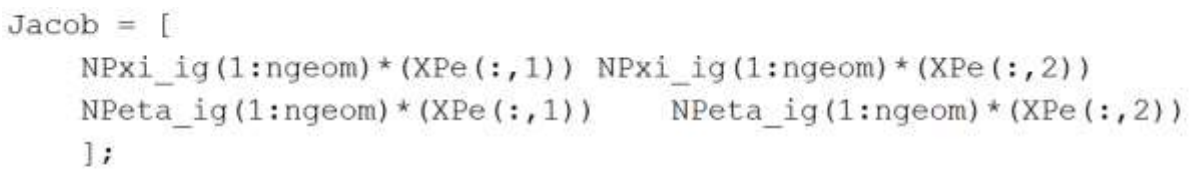
The first part of the assignment is to add the stabilization term for the Stokes problem. In order to do that, if stabilization is required, the program computes the L matrix and f_q force vector. The implementation has been the same as the given for the rest of matrices:



```
Le = Le + (NPx'*NPx+NPy'*NPy)*dvolu;
x_ig = N_ig(1:ngeom)*XPe;
f_igaus = SourceTerm(x_ig);
fe_q = fe_q + [NPx;NPy]'*f_igaus*dvolu;
```

Figure 1: Stabilization terms

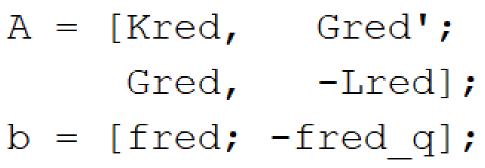
The main difference is that the gradient of the pressure is needed instead of the velocity one. For that, the computation of the Jacobian has been modified:



```
Jacob = [
    NPxi_ig(1:ngeom)*(XPe(:,1)) NPxi_ig(1:ngeom)*(XPe(:,2))
    NPeta_ig(1:ngeom)*(XPe(:,1))    NPeta_ig(1:ngeom)*(XPe(:,2))
    ];
```

Figure 2: Jacobian

The last step is the assembly of all the matrices. This has been done as follows:



```
A = [Kred,    Gred';
     Gred,    -Lred];
b = [fred; -fred_q];
```

Figure 3: Assembly of global system of equations

It is important to note that the last equation has been changed of sign in order to keep the symmetry of the system.

For the case of linear velocity and pressure, using quadrilateral elements, the results without stabilization were:

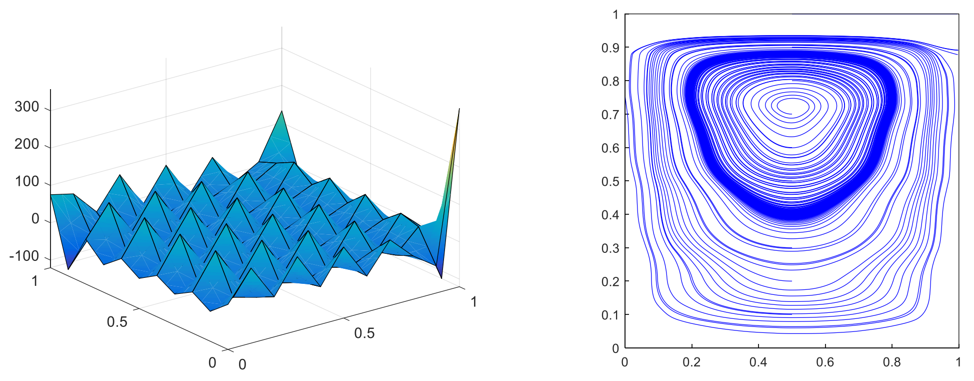


Figure 4: Pressure and streamlines for Stokes flow without stabilization terms

The same problem with the stabilization terms leads to more accurate results:

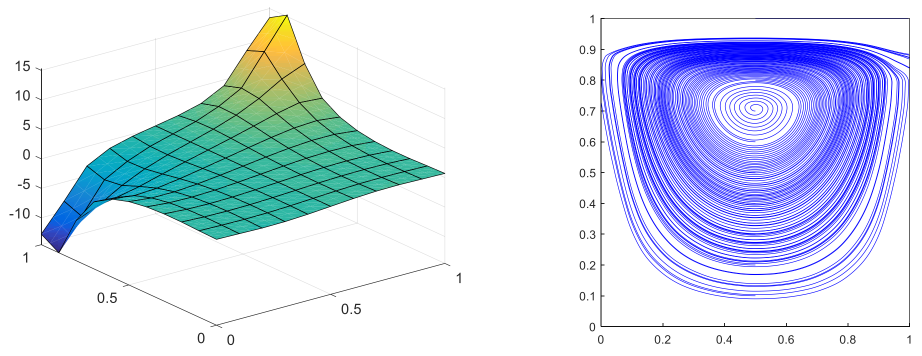
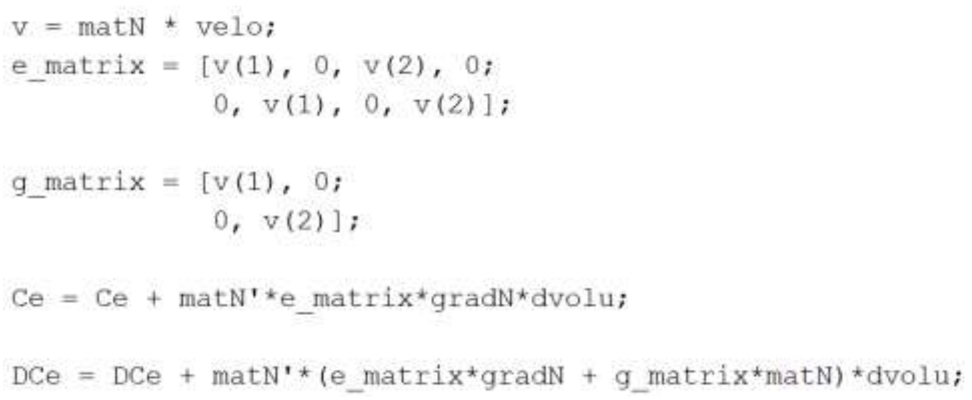


Figure 5: Pressure and streamlines for Stokes flow with stabilization terms

Navier-Stokes

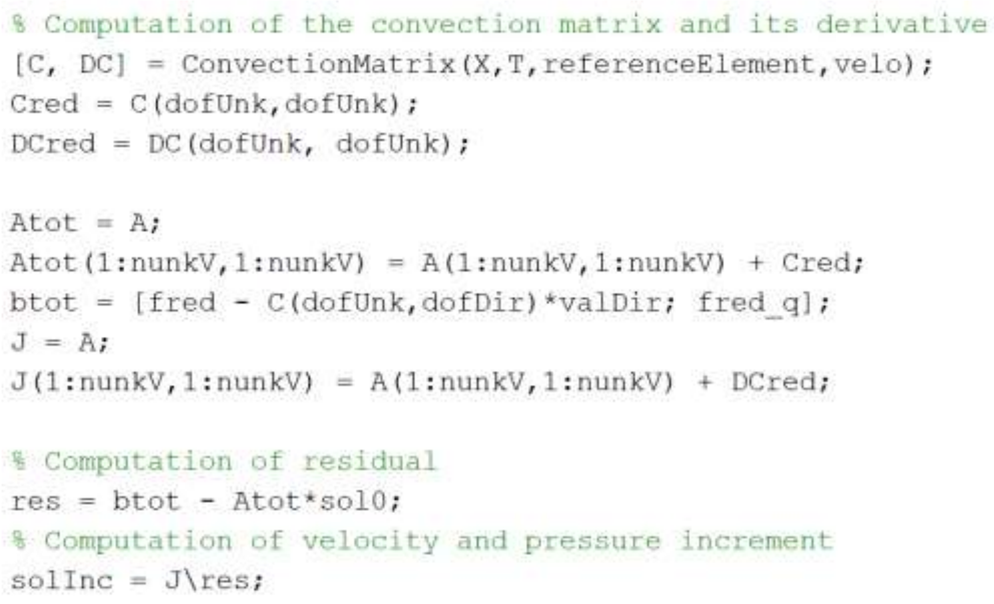
In this part of the assignment the two main tasks were to implement the computation of the convection matrix and its derivative. They have been implemented using a similar function as the one given for the assembly of the matrices K and G with the main difference of the non-linear term:



```
v = matN * velo;
e_matrix = [v(1), 0, v(2), 0;
            0, v(1), 0, v(2)];

g_matrix = [v(1), 0;
            0, v(2)];

Ce = Ce + matN'*e_matrix*gradN*dvolu;

DCe = DCe + matN'*(e_matrix*gradN + g_matrix*matN)*dvolu;
```

Figure 6: Assembly of the convection matrix

The last step is to implement the Newton-Raphson solver. This has been done modifying the Picard algorithm code:



```
% Computation of the convection matrix and its derivative
[C, DC] = ConvectionMatrix(X,T,referenceElement,velo);
Cred = C(dofUnk,dofUnk);
DCred = DC(dofUnk, dofUnk);

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

% Computation of residual
res = btot - Atot*sol0;
% Computation of velocity and pressure increment
solInc = J\res;
```

Figure 7: Newton-Raphson algorithm

The cavity has been simulated for a Reynolds number of 100. The domain has been discretized with a 10x10 mesh of quadrilateral linear elements with stabilization. The results are the following:

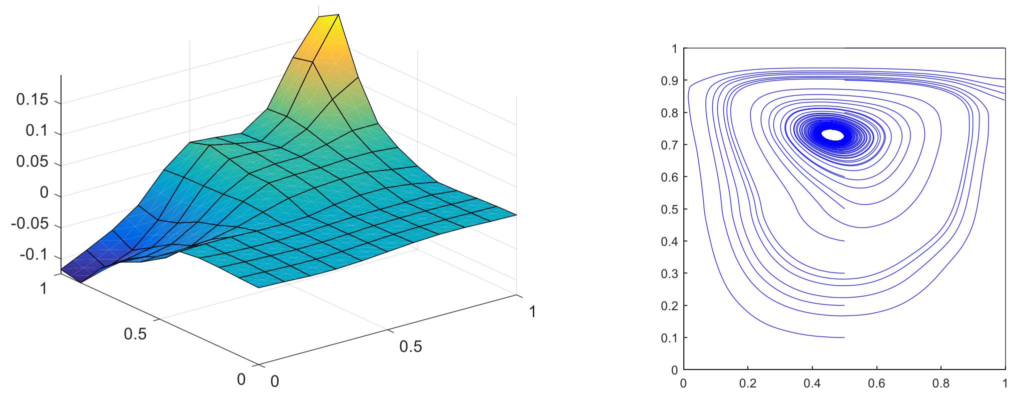


Figure 8: Pressure and streamlines for Navier-Stokes incompressible flow