FINITE ELEMENTS IN FLUIDS

Master of Science in Computational Mechanics/Numerical Methods

Spring Semester 2019

Rafel Perelló i Ribas

Laboratory: 4th session

Navier-Stokes transient flow

In this assignment, the Navier-Stokes transient equations have been solved using two different schemes: the implicit second order monolithic scheme and the algebraic splitting.

For the computation of the matrices to construct the system of equations the same function developed in the previous assignment is used. The only addition is the mass matrix. It has been used the consistent mass matrix as the problem to solve is not very costly computationally speaking. In a bigger problem it can be easily diagonalized summing all the contributions of each row to the diagonal.

Monolithic scheme

Implementation

The second order implicit method has been used. The problem to solve is the cavity of last assignment with initial velocity of 0. The algorithm is the following:

- In each time step the solution is initialized with the solution of the previous time step.

$$v_0^{n+1} = v^n$$

- The velocity $v_{n+\frac{1}{2}}$ is computed

$$v^{n+\frac{1}{2}} = \frac{1}{2}\left(v_k^{n+1} + v^n\right)$$

- Convection matrix $C$and its derivative $D(C \cdot u)/Du$ are computed using as velocity input $v_{n+\frac{1}{2}}$. The elemental convection matrix is computed as:

$$C_e = \sum_{i_{Gauss}} [\text{mat } \mathbf{N}]^{\mathrm{T}}\left(v_x[\text{grad}_{\text{x}}\ \mathbf{N}] + v_y[\text{grad}_{\text{y}}\mathbf{N}]\right) \cdot w$$

To compute the velocity at the Gauss points the trial functions are used:

$$v_x = av^e, v_y = bv^e$$

So the convection matrix is:

$$C_e = \sum_{i_{Gauss}} [\text{mat } \mathbf{N}]^{\mathrm{T}}\left([\text{grad}_{\text{x}}\ \mathbf{N}] \cdot (a \cdot u) + [\text{grad}_{\text{y}}\mathbf{N}] \cdot (b \cdot u)\right) \cdot w$$

Where:

$$a = N(x_{Gauss})\begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix}, \quad b = N(x_{Gauss})\begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix}$$

The derivative $D(C \cdot u)/Du$ is computed as follows:

$$\left[\frac{D(C \cdot u)}{Du}\right]_{ij} = \frac{DC_{ik}u_k}{Du_j} = \frac{DC_{ik}}{Du_j}u_k + C_{ik}\frac{Du_k}{Du_j}$$

The second term is just C_{ij}

$$C_{ik}\frac{Du_k}{Du_j} = C_{ik}\delta_{kj} = C_{ij}$$

The first term is:

$$\frac{DC_{ik}}{Du_j}u_k = \frac{D}{Du_j}\left(\sum_{i_{Gauss}} [\text{mat } \mathbf{N}]_{il}{}^{\mathrm{T}}\left([\text{grad}_{\mathrm{x}}\ \mathbf{N}]_{lk} \cdot (a_m \cdot u_m) + [\text{grad}_{\mathrm{y}}\mathbf{N}]_{lk} \cdot (b_m \cdot u_m)\right) \cdot w\right) \cdot u_k =$$

$$\sum_{i_{Gauss}} [\text{mat } \mathbf{N}]_{il}{}^{\mathrm{T}}\left([\text{grad}_{\mathrm{x}}\ \mathbf{N}]_{lk} \cdot (a_m \cdot \delta_{mj}) + [\text{grad}_{\mathrm{y}}\mathbf{N}]_{lk} \cdot (b_m \cdot \delta_{mj})\right) \cdot w \cdot u_k =$$

$$\sum_{i_{Gauss}} [\text{mat } \mathbf{N}]_{il}{}^{\mathrm{T}}\left([\text{grad}_{\mathrm{x}}\ \mathbf{N}]_{lk} \cdot a_j + [\text{grad}_{\mathrm{y}}\mathbf{N}]_{lk} \cdot b_j\right) \cdot w \cdot u_k$$

$$\frac{DC}{Du} \cdot u = \sum_{i_{Gauss}} [\text{mat } \mathbf{N}]^{\mathrm{T}}\left([\text{grad}_{\mathrm{x}}\ \mathbf{N}] \cdot u \cdot a^T + [\text{grad}_{\mathrm{y}}\mathbf{N}] \cdot u \cdot b^T\right) \cdot w$$

To calculate the derivative of the product $C[dofUnk, dofDir] \cdot valDir$ the same function is used. It has to be noted that $\frac{DC}{Du}$ is a third order tensor independent of u. The same function is used with input the vector $valDir$ in the same form as the velocity vector.

So the system of equations is:

$$f = \begin{bmatrix} f - C(dofUnk, dofDir) \cdot valDir \\ f_q \end{bmatrix}$$

$$A^{n+1} = \begin{bmatrix} \frac{1}{2}K + \frac{1}{\Delta t}M & G^T \\ G & -L \end{bmatrix}, A^n = \begin{bmatrix} \frac{1}{2}K - \frac{1}{\Delta t}M & 0 \\ 0 & 0 \end{bmatrix}$$

$$r = A^{n+1}x^{n+1} + A^n x^n + \begin{bmatrix} C & 0 \\ 0 & 0 \end{bmatrix} \cdot \frac{x^{n+1} + x^n}{2} - f$$

The Jacobian of the system is:

$$J = A^{n+1} + \frac{1}{2}\begin{bmatrix} C & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} \frac{DC}{Du} \cdot u - \frac{DC}{Du} \cdot valDir & 0 \\ 0 & 0 \end{bmatrix}$$

It has to be noted that all matrices have been reduced to its corresponding degrees of freedom.

Then, the next step is computed as:

$$\Delta x^{n+1} = -J^{-1}r$$

This loop is performed until convergence and then it is advanced to the next time step.

The convergence plot is the following:

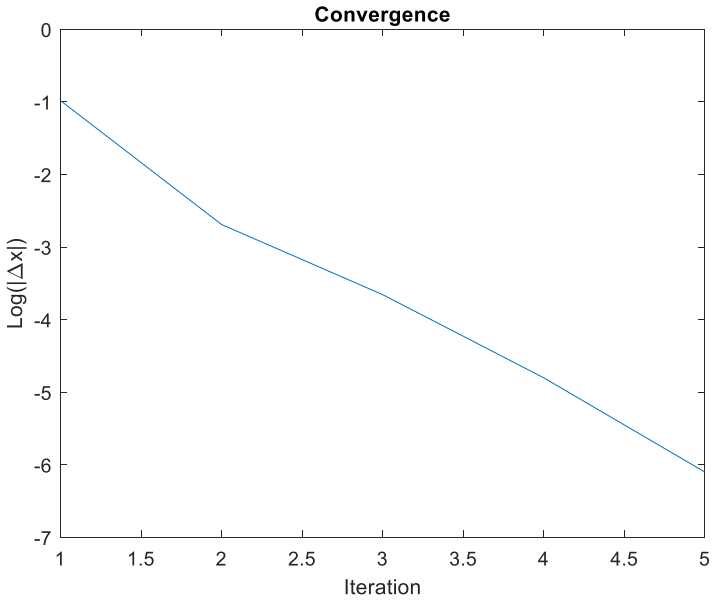

Figure 1: Convergence plot for the Newton-Raphson scheme

It is clearly seen that the convergence is not quadratic as expected. For that reason, a function has been coded to compute the numerical derivative of the residual and compare it with the Jacobian. In summary, the function assumes a new unknown $x_h^{n+1} = x^{n+1} + h$ and computes the residual with this unknown and performs a first order differentiation and compares it with the one computed with the Jacobian:

```
error = norm((resh - res) - J * h) / norm(h);
```

The error reduces linearly with the norm of h as expected but at about 10^{-4} it stagnates revealing that the Jacobian is not correct. Playing with the script, the conclusion is that the error is in the derivative of the force vector f. However, despite the efforts to solve it I have not been able to find the correct solution.

Results

It has been simulated the cavity with the same boundary conditions than in the previous assignment but with an initial velocity of 0. The Reynolds number is 100 and the density is of 5. It has been simulated 1 second divided in steps of 0.1s with 10 elements per edge. The results are the following:

- $t = 0.1s$

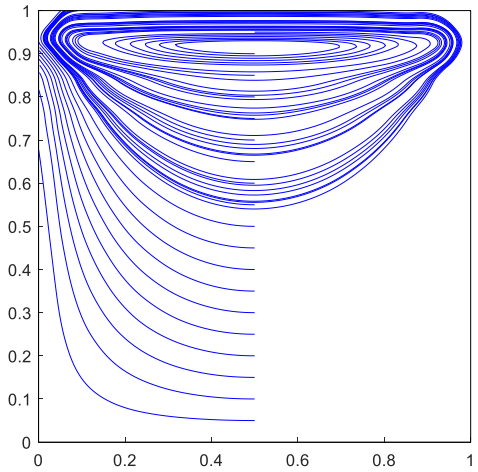

Figure 2: Streamlines at t=0.1s for monolithic scheme

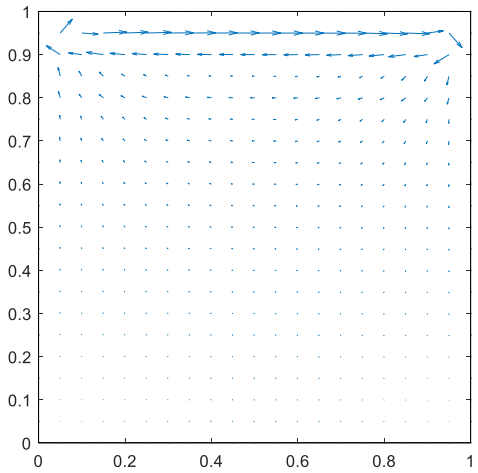

Figure 3: Velocity field at t=0.1s for monolithic scheme

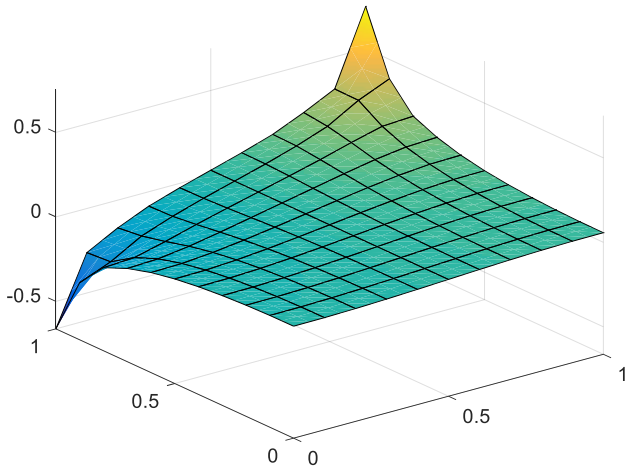

Figure 4: Pressure field at t=0.1s for monolithic scheme

- $t = 1s$

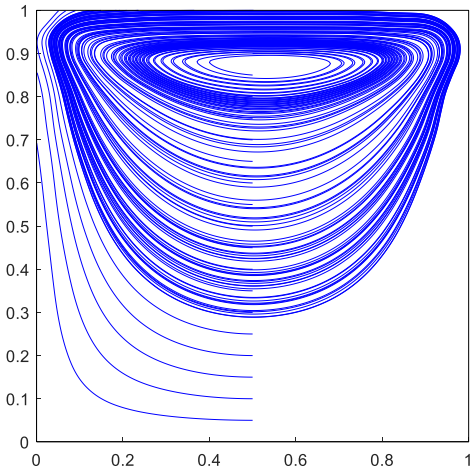

Figure 5:Streamlines at t=1s for monolithic scheme

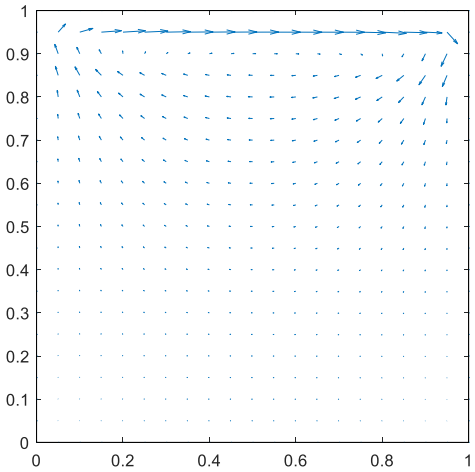

Figure 6: Velocity field at t=1s for monolithic scheme

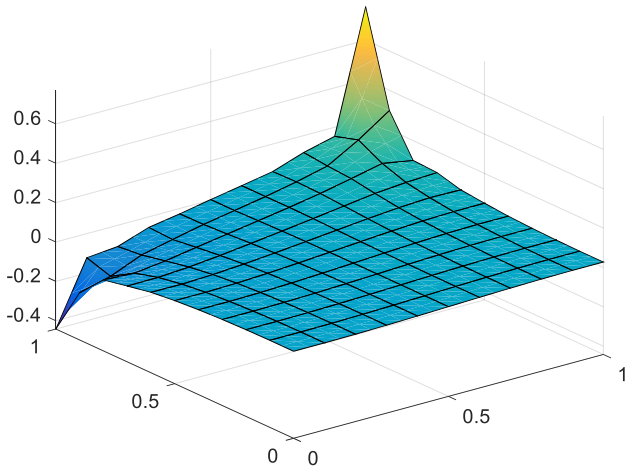

Figure 7: Pressure field at t=1s for monolithic scheme

Algebraic splitting

Implementation

For this implementation, most of the scheme and the computation of the matrices is exactly equal than in the monolithic scheme. First, two matrices are defined:

$$A_1 = \begin{bmatrix} \Delta t \cdot K + M & 0 \\ G & -GM^{-1}G' \end{bmatrix}, \qquad A_2 = M^{-1}G'$$

In each iteration first is computed the contribution to the force vector from the previous time step:

$$f_M = M \cdot v^n$$

Then, the solution is initialized with the one from the previous time step.

$$v_0^{n+1} = v^n$$

With this approximation, the convective matrix is computed and the first system is solved:

$$\begin{bmatrix} \widehat{v^{n+1}} \\ \widehat{p^{n+1}} \end{bmatrix}^{k+1} = \left(A_1 + \begin{bmatrix} C \cdot \Delta t & 0 \\ 0 & 0 \end{bmatrix} \right)^{-1} \cdot \left(f + \begin{bmatrix} f_M \\ 0 \end{bmatrix} \right)$$

Where the force vector f takes into account the contributions of the Dirichlet Boundary Condition.

Then:

$$p^{n+1} = \widehat{p^{n+1}}, \qquad v^{n+1} = \widehat{v^{n+1}} - A_2 \cdot p^{n+1}$$

After that, the iteration is complete and it starts again. This scheme is a fixed point iteration and for the problem solved converges in 5 iterations to a tolerance of 10^{-6}.

Results

The simulation performed is the same than in the previous algorithm. The results are the following:

- $t = 0.1s$

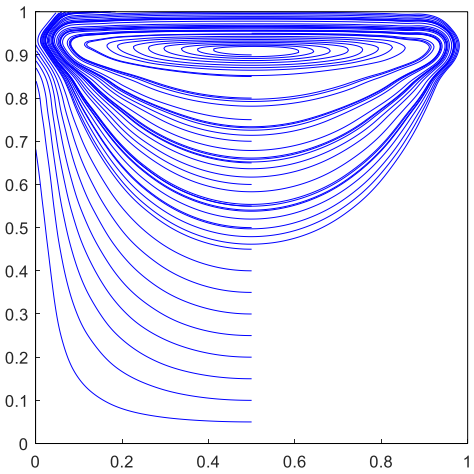

Figure 8:Streamlines at t=0.1s for algebraic splitting scheme

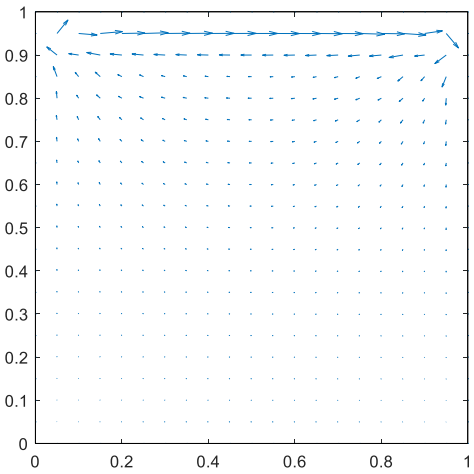

Figure 9: Velocity field at t=0.1s for algebraic splitting scheme

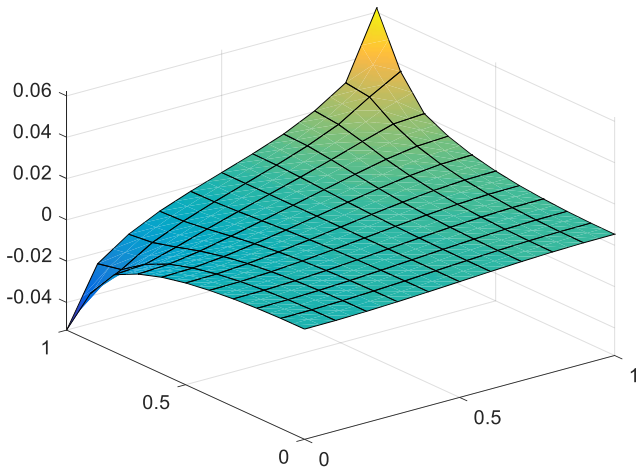

Figure 10: Pressure field at t=0.1s for algebraic splitting scheme

- $t = 1s$

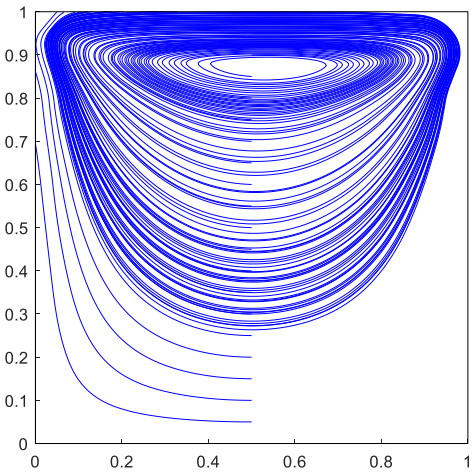

Figure 11: Streamlines at t=1s for algebraic splitting scheme

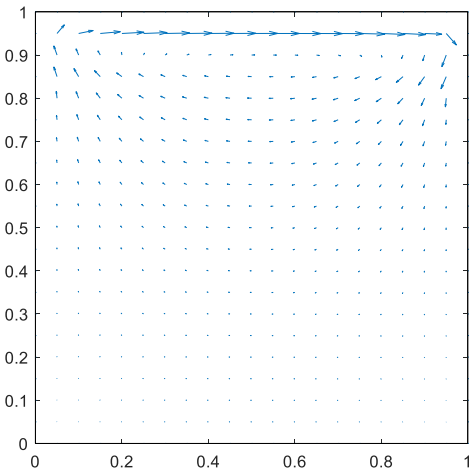

Figure 12: Velocity field at t=1s for algebraic splitting scheme

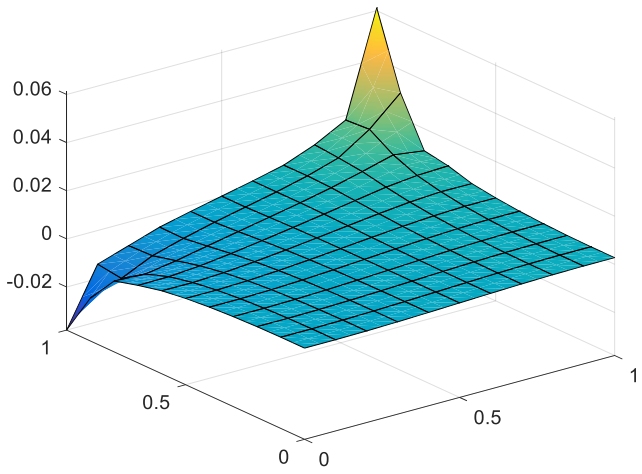

Figure 13: Pressure field at t=1s for algebraic splitting scheme

Conclusions

Both methods yield to the same results up to numerical errors. So, we conclude that the implementation of the schemes is correct with the mentioned error in the computation of the Jacobian in the Newton-Raphson solver.