Unsteady Navier-Stokes problem

Arthur Lustman

April 25, 2019

1 Implicit Second order monolithic scheme

1.1 Mathematical Description

The monolithic schemes mathematical expression, starting from the theta method, where $\theta = 1/2$

$$\frac{v^{n+1}-v^n}{\Delta t} - \frac{1}{2}\left(f-(\overline{v}\cdot\nabla)\overline{v}+\nu\nabla^2\overline{v}\right)^{n+1} + \frac{1}{2}\nabla p^{n+1} = \frac{1}{2}\left(f-(\overline{v}\cdot\nabla)\overline{v}+\nu\nabla^2\overline{v}\right)^{n} \tag{1}$$

$$\nabla\cdot\overline{v}^{n+1} = 0 \tag{2}$$

The gradient of the velocity and the pressure are evaluated at t^{n+1}. The first equation can be rewritten

$$\underbrace{\frac{v^{n+1}}{\Delta t}}_{\frac{1}{\Delta t}\boldsymbol{M}} + \frac{1}{2}\underbrace{(\overline{v}^{n+1}\cdot\nabla)\overline{v}^{n+1}}_{-\boldsymbol{C}^\top(\overline{v}^{n+1})} - \frac{1}{2}\underbrace{\nu\nabla^2\overline{v}^{n+1}}_{\boldsymbol{K}} + \frac{1}{2}\nabla p^{n+1} = \frac{1}{2}f^n + \frac{1}{2}f^{n+1} - \frac{1}{2}\underbrace{(\overline{v}^{n}\cdot\nabla)\overline{v}^{n}}_{\boldsymbol{C}^\top\overline{v}^n} + \frac{1}{2}\underbrace{\nu\nabla^2\overline{v}^{n}}_{\boldsymbol{K}} + \underbrace{\frac{1}{\Delta t}v^n}_{\frac{1}{\Delta t}\boldsymbol{M}} \tag{3}$$

Note that the under-braced matrices are here to display from which terms those matrices comes from and they do not embrace the exact value. They are the outcome of a finite element discretization which is not described here. They have already been explained during the course but are still rewritten here

$$\boldsymbol{K} \leftarrow \int_\Omega [\text{grad}\boldsymbol{N}]^\top [\text{grad}\boldsymbol{N}] d\Omega \tag{4}$$

$$\boldsymbol{G} \leftarrow -\int_\Omega \hat{\boldsymbol{N}}^\top \boldsymbol{D} d\Omega \tag{5}$$

$$\boldsymbol{f} \leftarrow \int_\Omega \boldsymbol{N}^\top \boldsymbol{f} d\Omega \tag{6}$$

$$\boldsymbol{M} \leftarrow \int_\Omega \boldsymbol{N}^\top \boldsymbol{N} d\Omega \tag{7}$$

$$\boldsymbol{C}(\overline{v}^n) \leftarrow \sum_e \int_\Omega \boldsymbol{N}^\top \begin{bmatrix} v_x^n & 0 & v_y^n & 0 \\ 0 & v_x^n & 0 & v_y^n \end{bmatrix} [\text{grad}\boldsymbol{N}] \tag{8}$$

This leaves us with the following system of equation

$$\begin{bmatrix} \frac{1}{\Delta t}\boldsymbol{M} - \frac{1}{2}\boldsymbol{C}^\top(\overline{v}^{n+1}) + \frac{1}{2}\boldsymbol{K} & \boldsymbol{G}^\top \\ \boldsymbol{G} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \overline{v}^{n+1} \\ p^{n+1} \end{bmatrix} = \begin{bmatrix} f^n + \frac{1}{2}\boldsymbol{C}(\overline{v}^n)\overline{v}^n - \frac{1}{2}\boldsymbol{K}\overline{v}^n + \frac{1}{\Delta t}\boldsymbol{M}\overline{v}^n \\ 0 \end{bmatrix} \tag{9}$$

Or, in a simple way

$$\begin{bmatrix} \boldsymbol{A}(\overline{v}^{n+1}) & \boldsymbol{G}^\top \\ \boldsymbol{G} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \overline{v}^{n+1} \\ p^{n+1} \end{bmatrix} = \begin{bmatrix} b^n \\ 0 \end{bmatrix} \tag{10}$$

1.2 Newton-Raphson solution

This previous non-linear system of equation is computed using Newton-Raphson. The residual $\boldsymbol{r}$ is described as

$$\boldsymbol{r}^n = \begin{bmatrix} \boldsymbol{A}(\overline{v}^n)\overline{v}^n + \boldsymbol{G}^\top p^n - b \\ \boldsymbol{G}\overline{v}^n \end{bmatrix} \tag{11}$$

For which the Jacobian, which is the derivative of the residual in terms of the unknows is necessary to compute the solution

$$\boldsymbol{J}^n = \begin{bmatrix} \frac{d\boldsymbol{r}_1^n}{d\overline{v}} & \frac{d\boldsymbol{r}_1^n}{dp} \\ \frac{d\boldsymbol{r}_2^n}{d\overline{v}} & \frac{d\boldsymbol{r}_2^n}{dp} \end{bmatrix} = \begin{bmatrix} \frac{d\boldsymbol{r}_1^n}{d\overline{v}} & \boldsymbol{G}^\top \\ \boldsymbol{G} & 0 \end{bmatrix} \tag{12}$$

For which the expression of the component $\frac{d\boldsymbol{r}_1^n}{d\overline{v}}$ is equal to

$$\boldsymbol{A}(\overline{v}^{n+1}) + \frac{\partial \boldsymbol{C}}{\partial \overline{v}}\overline{v}^{n+1} \tag{13}$$

Where the partial derivative is computed using directional derivatives

$$D(\overline{w}(\overline{v}\cdot\nabla)\overline{v})[\delta\overline{v}] = \underbrace{\overline{w}(\delta\overline{v}\cdot\nabla)\overline{v}}_{\frac{\partial \boldsymbol{C}}{\partial \overline{v}}(\overline{v})} + \underbrace{\overline{w}(\overline{v}\cdot\nabla)\delta\overline{v}}_{\boldsymbol{C}(\overline{v})} \tag{14}$$

$$\frac{\partial \boldsymbol{C}}{\partial \overline{v}}(\overline{v}) \leftarrow \sum_e \int_\Omega \boldsymbol{N}^\top \begin{bmatrix} \frac{\partial v_x}{\partial x} & \frac{\partial v_x}{\partial y} \\ \frac{\partial v_y}{\partial x} & \frac{\partial v_y}{\partial y} \end{bmatrix} \boldsymbol{N} d\Omega \tag{15}$$

The solution is then computed as

$$\begin{bmatrix} \overline{v}^{n+1} \\ p^{n+1} \end{bmatrix} = \begin{bmatrix} \overline{v}^n \\ p^n \end{bmatrix} - (\boldsymbol{J}^n)^{-1}\boldsymbol{r}^n \tag{16}$$

A characteristic of the Newton-Raphson convergence is the quadratic convergence. It should be observed in the results when looking at the results.

1.3 Code Implementation

The implementation of this code is pretty straightforward, if taken as a start the code from the previous homework, Stokes over a cavity flow problem. Two additional files computing the Mass matrix and the partial derivative of $\boldsymbol{C}$ are needed and implemented. Though most of their lines is common to the file `StokesSystem` who was computing the matrices $\boldsymbol{K}$, $\boldsymbol{G}$ and f. For this reason, most of the code is not written, only the loops (over the number of elements and the number of gauss points) are stated and the modified lines.

Mass Matrix

```
function M = CreateMass(X,T,referenceElement)
[...]
M = zeros(ndofV,ndofV);

for ielem = 1:nElem
    [...]
    Me = zeros(nedofV,nedofV);
    for ig = 1:ngaus
        [...]
        Me = Me + Ngp'*Ngp*dvolu;
    end
    M(Te_dof, Te_dof) = M(Te_dof, Te_dof) + Me;
end
```

Where `Ngp` are the shape functions related to the velocity.

dC/dv

```
function dC = dCdv(X,T,referenceElement,velo)
% velo: velocity values at the nodes
[...]
dC = zeros(ndofV,ndofV);

for ielem = 1:nElem
    [...]
    dCe = zeros(nedofV,nedofV);
    % Element matrices
    Ve = velo(Te,:);
    for ig = 1:ngaus
        [...]
        gradV = [nx*Ve(:,1) ny*Ve(:,1);nx*Ve(:,2) ny*Ve(:,2)];
        dCe = dCe + Ngp'*gradV*Ngp*dvolu;
    end
    % Assemble the element matrices
    dC(Te_dof, Te_dof) = dC(Te_dof, Te_dof) + dCe;

end
```

These two files are needed for the Newton-Raphson solution method. The structure of the code is given here in order to fully understand the solution procedure. The basic structure of the problem is defined

$$\begin{bmatrix} \frac{1}{2}\boldsymbol{K} + \frac{1}{\Delta t}\boldsymbol{M} + \texttt{First} & \boldsymbol{G}^\top \\ \boldsymbol{G} & 0 \end{bmatrix} \begin{bmatrix} v^{k+1} \\ p^{k+1} \end{bmatrix} = \begin{bmatrix} f^k + \texttt{Second} \\ 0 \end{bmatrix} \tag{17}$$

Where the variables `First` and `Second` are computed within each k iteration inside a time step.

$$\texttt{First} = -\frac{1}{2}\boldsymbol{C}^\top(\overline{v}^k) \tag{18}$$

$$\texttt{Second} = \frac{1}{2}\boldsymbol{C}(\overline{v}^k)\overline{v}^k - \frac{1}{2}\boldsymbol{K}\overline{v}^k + \frac{1}{\Delta t}\boldsymbol{M}\overline{v}^k \tag{19}$$

Newton-Raphson

```
while time step < end time step
    while tolerance is not met
        compute C, dC/dV with v^k
        compute First and Second variables
        add their contribution to the left matrix and right vector
                    of equation (17)

        compute residual^k and Jacobian
        solve equation (16) to get v^k+1 and p^k+1
        check if residual < tolerance
    end
    update solution [v;p]^k=[v;p]^k+1
end
```

1.4 Results

The Navier-Stokes problems is solved over the problem of cavity flow from the previous assignment. The parameter for the time integration is the number of time steps nt, the time step Δt and the

tolerance, which is chosen to be 10^{-6}. The solution has been computed two time with different parameters :

- $\Delta t = 0.01s$ with 10 time steps
- $\Delta t = 0.1s$ with 10 time steps

The convergence is assured by the implicit scheme but a small Δt should provide us precise results and a small nt should assure us that it won't be taking too long the reach the solution. The graphical results are displayed in the following figures as well as the convergence of the residual.

The y axis of the convergence figures are in logarithmic scale. The convergence shows spike patterns because approximately 2-3 iterations are necessary for each time step. The rate of convergence is around ≈ 2, meaning the convergence is quadratic as it should be for a Newton-Raphson algorithm.

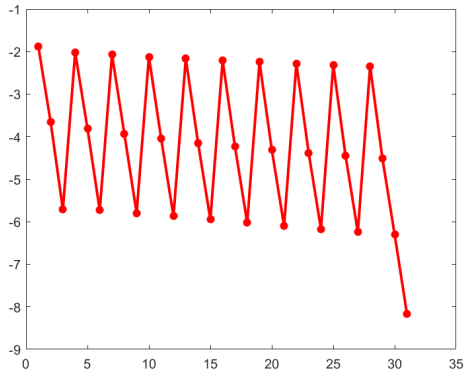

Figure 1: Convergence for $\Delta t = 0.01$

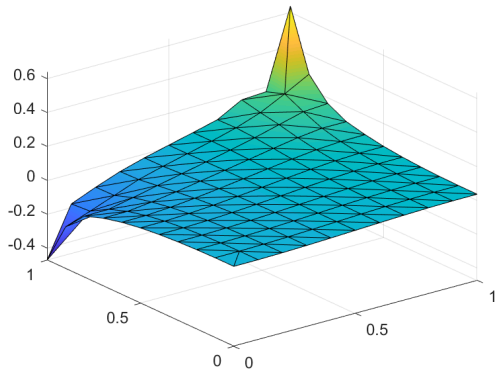

Figure 2: Graph of the pressure for $\Delta t = 0.01$

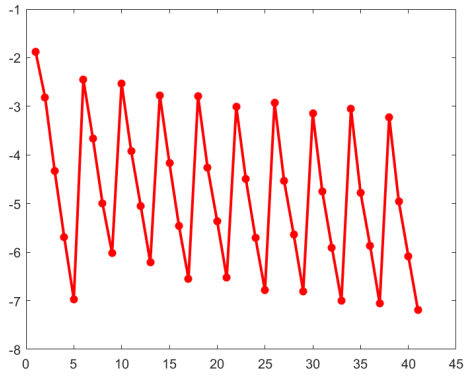

Figure 3: Convergence for $\Delta t = 0.1$

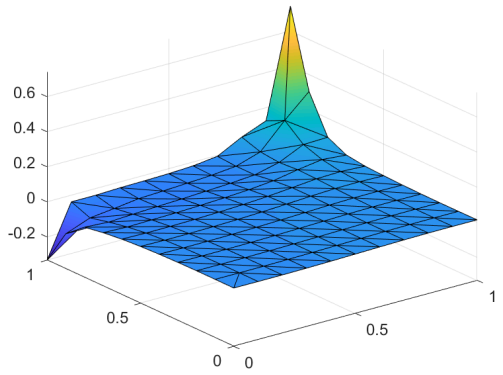

Figure 4: Graph of the pressure for $\Delta t = 0.1$

2 Algebraic splitting

The Navier-Stokes equations yields a system of equations to be solved at each time-step.

$$\begin{pmatrix} \boldsymbol{A} & \boldsymbol{G}^\top \\ \boldsymbol{G} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} u^{n+1} \\ p^{n+1} \end{pmatrix} = \begin{pmatrix} f^{n+1} \\ h^{n+1} \end{pmatrix} \tag{20}$$

With $\boldsymbol{A} = \boldsymbol{M} + \Delta t \left(\boldsymbol{K} + \boldsymbol{C}(v^{n+1})\right)$. Thought this system of equation could be solved directly, using the schur complement:

$$\left(\boldsymbol{G}\boldsymbol{A}^{-1}\boldsymbol{G}^\top\right) p^{n+1} = \boldsymbol{G}\boldsymbol{A}^{-1} f^{n+1} - h^{n+1} \tag{21}$$

$$\boldsymbol{A} v^{n+1} = f^{n+1} - \boldsymbol{G}^\top p^{n+1} \tag{22}$$

But this type of solution tends to be quite computationally expensive as we have to compute the inverse of a sparse matrix $\boldsymbol{A}$ which is dense.

The first step of this solution doesn't need to use the inverse of the matrix in the case of a direct solver, it remains cheap in case of rationally small matrices.

2.1 LU Step and A approximation

One way to evade this problem is to perform the algebraic splitting of the system of equation 20 in two triangular LU matrices. The process is defined on the slides and in the reference book page 304 and 305.

In this case the solution is divided in three simpler steps:

$$\boldsymbol{A}\hat{v}^{n+1} = f^{n+1} \tag{23}$$

$$\left(\boldsymbol{G}\boldsymbol{A}^{-1}\boldsymbol{G}^\top\right) \hat{p}^{n+1} = h^{n+1} - \boldsymbol{G}\hat{v}^{n+1} \tag{24}$$

$$\boldsymbol{A} v^{n+1} = \boldsymbol{A}\hat{v}^{n+1} - \boldsymbol{G}^\top p^{n+1} \tag{25}$$

The computational time is reduced by using an approximation of $\boldsymbol{A}^{-1}$, which comes from ignoring the inverse of all the other terms, except of the mass matrix $\boldsymbol{M}$. Since it is a block diagonal matrix, it is easy to inverse and keeps its sparsity. The three steps are now defined as

$$\boldsymbol{A}\hat{v}^{n+1} = f^{n+1} \tag{26}$$

$$\left(\boldsymbol{G}\boldsymbol{M}^{-1}\boldsymbol{G}^\top\right) \hat{p}^{n+1} = h^{n+1} - \boldsymbol{G}\hat{v}^{n+1} \tag{27}$$

$$\boldsymbol{M} v^{n+1} = \boldsymbol{M}\hat{v}^{n+1} - \boldsymbol{G}^\top p^{n+1} \tag{28}$$

2.2 Code Implementation

Since the first equation of the solver is non linear (equation (23)), there is a need of Newton-Raphson or Picard method in order to solve it. This is first performed only for the velocity, which become the intermediate velocity $\hat{v}$. Since the Newton-Raphson code is precisely described in the previous section, there is no need to cover it again. The two other equations (24) and (25) are straightforward to implement.

Approximation A and Algebraic splitting

```
while time step < end time step
    while tolerance is not met
        [Newton-Raphson for v_hat and reduced system]
    end
    S = G*inv(M)*G';
    New_Pressure = -S\G*v_hat;
    New_Velocity = M\(M*v_hat - G'*New_Pressure);
    update solution [v;p]^k=[v;p]^k+1
end
```

2.3 Results

The code can reach a stable solution but only for a reasonable *Deltat*. The quadratic convergence is met when $\Delta t = 0.01$ as it can be seen in the figure 5, but it is slightly too low with the time updates when $\Delta t = 0.1$. Yet the only trouble is the quadratic convergence in case Δt is big since the flow seem very good.

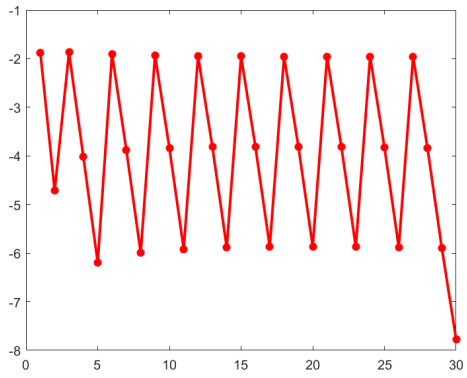

Figure 5: Convergence for $\Delta t = 0.01$

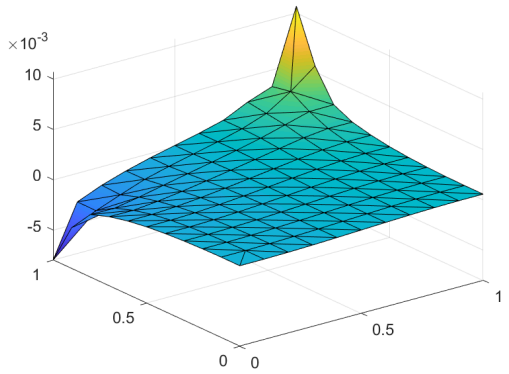

Figure 6: Graph of the pressure for $\Delta t = 0.01$

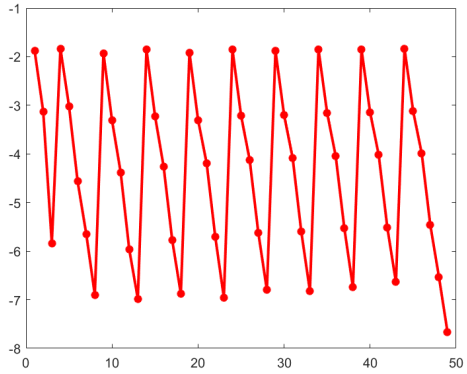

Figure 7: Convergence for $\Delta t = 0.1$

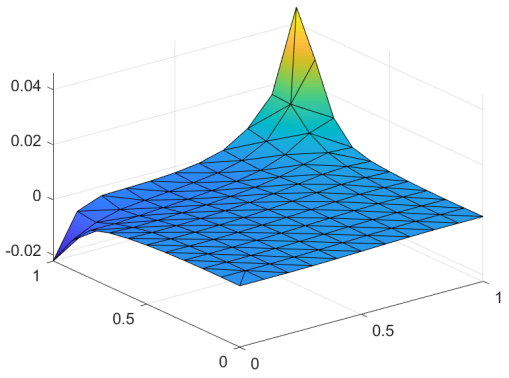

Figure 8: Graph of the pressure for $\Delta t = 0.1$

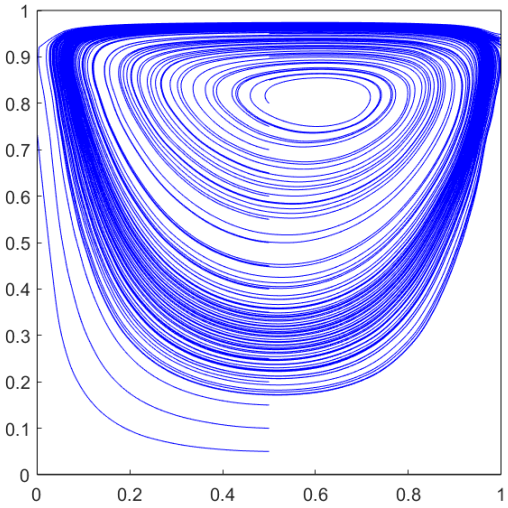

Figure 9: Flow for $\Delta t = 0.1$