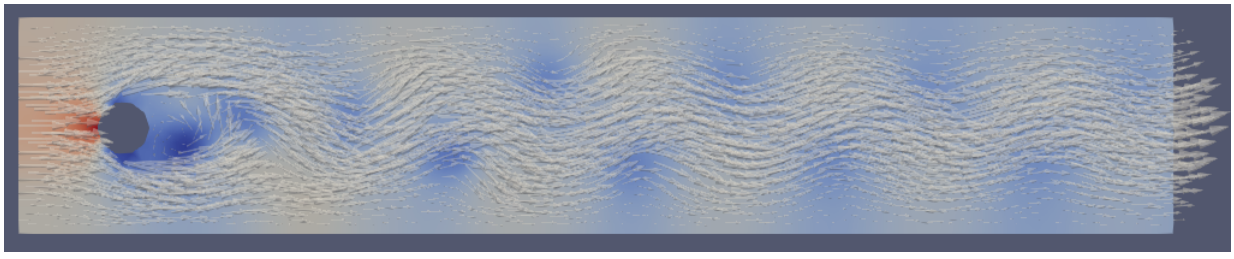
Faculty of Sciences - Course 2019/2020

PDEs, modeling and simulation

6th of December 2019

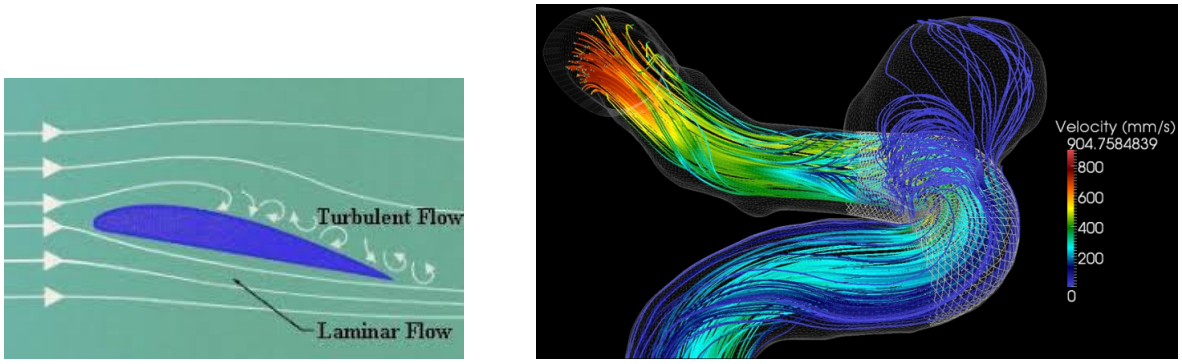
Leroux Louis-Clément



Contents

1 Introduction

The Navier-Stokes equations are the equations used to simulate viscous fluid substances. They describe the physics, and they are used in various scientific and engineering areas. We use them for example to model air flow around a wing in the aeronautics industry, water flow in a pipe in engineering, but also simulate the weather or ocean currents. They are also used in medicine to study the blood flow. In fact, in every area where we need to simulate a flow.



(a) Flow around a wing

(b) Blood flow

Figure 1: Simulations using Navier-Stokes equations

The subject of this document is to do simple simulations in two dimensions using Navier-Stokes equations. For that, I will first present Chorin's method, which is a method to solve these equations. Then, I will explain an improvement of this method, called IPC. It gives improved accuracy compared to the original scheme at little extra cost. And finally I will simulate a flow using FEniCS (it's an open-source computing platform for solving partial differential equations).

2 Chorin's method

Navier-Stokes equations are a system of two equations:

$$\frac{\partial u}{\partial t} + (u \cdot \nabla)u = -\frac{1}{\rho}\nabla p + D\Delta u \tag{1}$$

$$\nabla \cdot u = 0 \tag{2}$$

(1) is the conservation of momentum. That is Newton's second law.
(2) is the equation of continuity (mass balance).

Notations:

- u: velocity
- p: pressure
- ρ: density
- D: viscosity
- v: vector-valued test function
- q: scalar-valued test function
- $\nabla\cdot$: divergence
- Δ: Laplacian

Due to the resources needed to solve these equations, the usage of direct methods like finite element is not feasible and we need to split the problem.

Chorin's method is a splitting approach, which consists of considering the velocity and pressure forces separately.
The first step of Chorin's method is to compute a tentative velocity, by ignoring the pressure.

$$\begin{aligned}
\frac{\partial u}{\partial t} + (u \cdot \nabla)u &= -\frac{1}{\rho}\nabla p + D\Delta u^n \\
\rightsquigarrow \frac{u^* - u^n}{\Delta t} &= -(u^n \cdot \nabla)u^n - \frac{1}{\rho}\nabla p + D\Delta u^n \\
\implies \frac{u^* - u^n}{\Delta t} &= -(u^n \cdot \nabla)u^n + D\Delta u^n \\
\implies u^* &= u^n + \Delta t(-u^n \cdot \nabla u^n + D\Delta u^n)
\end{aligned}$$

Here we have the expression of our tentative velocity u^*. Now we have it, we can express the velocity of our next step: u^{n+1}.

$$\frac{u^{n+1}-u^n}{\Delta t} = -(u^n \cdot \nabla)u^n - \frac{1}{\rho}\nabla p^{n+1} + D\Delta u^n$$
$$\frac{u^{*}-u^n}{\Delta t} = -(u^n \cdot \nabla)u^n + D\Delta u^n$$
$$\implies u^{n+1} = u^{*} - \frac{\Delta t}{\rho}\nabla p^{n+1}$$

Then, we have to compute the pressure of the next step. For that, we use the divergence and the fact that $\nabla \cdot u^{n+1} = 0$. Thus we have:

$$\nabla \cdot u^{n+1} = \nabla \cdot u^{*} - \frac{\Delta t}{\rho}\Delta p^{n+1} = 0$$
$$\implies \Delta p^{n+1} = \frac{\rho}{\Delta t}\nabla \cdot u^{*}$$

Here we have the expression of the next pressure depending of the tentative velocity that we have just calculated.
The third and final step is to compute the next velocity, and we have the value of every variable we need.

$$u^{n+1} = u^{*} - \frac{\Delta t}{\rho}\nabla p^{n+1}$$

Variational formulation: The variational formulation also known as weak formulation allows to find in a fast and simple way the solution of PDEs. To use FEniCS, we have to use our equations in variational formulation.

$$\frac{u^{*}-u^n}{\Delta t} = -(u^n \cdot \nabla)u^n - \frac{1}{\rho}\nabla p + D\Delta u^n$$
$$\implies \int_\Omega \frac{u^{*}-u^n}{\Delta t} v dx = -\int_\Omega (u^n \cdot \nabla)u^n v dx + \int_\Omega D\Delta u^n v dx$$
$$\implies \int_\Omega \frac{u^{*}-u^n}{\Delta t} v dx = -\int_\Omega (u^n \cdot \nabla)u^n v dx - \int_\Omega D\nabla u^n \nabla v dx$$

$$\Delta p^{n+1} = \frac{\rho}{\Delta t}\nabla \cdot u^{*}$$
$$\implies \int_\Omega \Delta p^{n+1} q dx = \int_\Omega \frac{\rho}{\Delta t}\nabla u^{*} q dx$$
$$\implies \int_\Omega \nabla p^{n+1} \nabla q dx = -\int_\Omega \frac{\rho}{\Delta t}\nabla u^{*} q dx$$

$$u^{n+1} = u^{*} - \frac{\Delta t}{\rho}\nabla p^{n+1}$$
$$\implies \int_\Omega u^{n+1} v dx = \int_\Omega u^{*} v dx - \int_\Omega \frac{\Delta t}{\rho}\nabla p^{n+1} v dx$$

3 IPCS (Incremental Pressure Correction Scheme)

This method is similar to Chorin's method, but it gives improved accuracy compared to the original one. Actually, instead of ignoring completely the previous pressure like Chorin's Method do, we compute the tentative velocity with the previous pressure.
We get for the variational problem for the first step:

$$\int_\Omega \rho \frac{u^* - u^n}{\Delta t} v dx + \int_\Omega \rho u^n \cdot \nabla u^n v dx + \int_\Omega \sigma(u^{n+\frac{1}{2}}, p^n) : \epsilon(v) dx + \int_{\partial\Omega} p^n n v dx - \int_{\partial\Omega} D\nabla u^{n+\frac{1}{2}} n dv = \int_\Omega f^{n+1} v dx$$

Notations:

- $\sigma(u, p)$: stress tensor
- $\epsilon(u)$: strain-rate tensor

We use the notation $u^{n+\frac{1}{2}}$ for the value of u at the midpoint of the interval.

$$u^{n+\frac{1}{2}} = \frac{u^* + u^n}{2}$$

Now that we have our tentative velocity u^* based on the previous pressure, we can compute the new pressure p^{n+1}.

$$\int_\Omega \nabla p^{n+1} \cdot \nabla q dx = \int_\Omega \nabla p^n \cdot \nabla q dx - \frac{1}{\Delta t} \int_\Omega \nabla \cdot u^* q dx$$

We can note here that instead of the vector-valued test function v we used for the velocity, we use here for the pressure a scalar-valued test function q.

This step comes from, like for Chorin's method, with the fact that:

$$\frac{u^{n+1} - u^*}{\Delta t} + \nabla p^{n+1} - \nabla p^n = 0$$

$$\nabla \cdot u^{n+1} = 0$$

Finally, we have the expression of the last step: computing the new velocity.

$$\int_\Omega u^{n+1} v dx = \int_\Omega u^* v dx - \Delta t \int_\Omega \nabla (p^{n+1} - p^n) v dx$$

4 FEniCS implementation

The first simulation we want to do is a basic one: in a channel, with a velocity inflow on one side, we will simulate the velocity and pressure in this channel. We will add one obstacle, a cylinder in the center, to see the path of the flow around this cylinder. Before showing the result, I will explain the code.

For this implementation, we will need to define two function spaces. One is for the velocity and the other one is for the pressure.

```
# Define function spaces
V = VectorFunctionSpace(mesh, 'P', 2) #vector field
Q = FunctionSpace(mesh, 'P', 1) #scalar field
```

Figure 2: Function spaces

V is a vector-valued function space for the velocity. Q is a scalar-valued function space for the pressure. So we can use piecewise quadratic elements for the velocity and piecewise linear elements for the pressure.

Then, we have to define two trial functions and two test functions, as we have two different function spaces.

```
# Define trial and test functions
u = TrialFunction(V)
v = TestFunction(V)
p = TrialFunction(Q)
q = TestFunction(Q)
```

Figure 3: Trial and test functions

Then we can create our channel and our cylinder, or every form we want. We can apply the mesh on this domain.

```
# Create mesh
channel = Rectangle(Point(0, 0), Point(2.2, 0.41))
cylinder = Circle(Point(0.2, 0.2), 0.05)
domain = channel - cylinder
mesh = generate_mesh(domain, 64) #64 cells across the channel length
```

Figure 4: Domain

Now we can define the boundaries and their conditions. For the boundaries, we precise the boundary of the domain. For the cylinder, as it's not straight lines, we select all the boundaries in a rectangle where is the cylinder.
For the conditions, we want a velocity of 0 on the wall and the outflow. For the inflow, we define a function depending of the channel's height. The inflow should be 0 on the top and bottom, and then increase to the maximum in the center of the channel.
All the boundary conditions are added in a list to easily access them later.

```
# Define boundaries
inflow   = 'near(x[0], 0)'
outflow  = 'near(x[0], 2.2)'
walls    = 'near(x[1], 0) || near(x[1], 0.41)'
cylinder = 'on_boundary && x[0]>0.1 && x[0]<0.3 && x[1]>0.1 && x[1]<0.3'

# Define inflow profile
inflow_profile = ('4.0*1.5*x[1]*(0.41 - x[1]) / pow(0.41, 2)', '0')

# Define boundary conditions
bcu_inflow = DirichletBC(V, Expression(inflow_profile, degree=2), inflow)
bcu_walls = DirichletBC(V, Constant((0, 0)), walls)
bcu_cylinder = DirichletBC(V, Constant((0, 0)), cylinder)
bcp_outflow = DirichletBC(Q, Constant(0), outflow)
bcu = [bcu_inflow, bcu_walls, bcu_cylinder]
bcp = [bcp_outflow]
```

Figure 5: Boundaries and their conditions

We will now define our variational problems, one for each of the three steps in the IPC scheme. Firstly, we define our constants:

```
# Define expressions used in variational forms
U  = 0.5*(u_n + u)
n  = FacetNormal(mesh)
f  = Constant((0, 0))
k  = Constant(dt)
mu = Constant(mu)
rho = Constant(rho)
```

Figure 6: Constants

We can now compute our first step: the tentative velocity. As notations, we will write u for the tentative velocity, and $u\_$ for the previous one.
Then, we compute the new pressure, and finally the new velocity:

```
# Define variational problem for step 1
F1 = rho*dot((u - u_n) / k, v)*dx \
   + rho*dot(dot(u_n, nabla_grad(u_n)), v)*dx \
   + inner(sigma(U, p_n), epsilon(v))*dx \
   + dot(p_n*n, v)*ds - dot(mu*nabla_grad(U)*n, v)*ds \
   - dot(f, v)*dx
a1 = lhs(F1)
L1 = rhs(F1)

# Define variational problem for step 2
a2 = dot(nabla_grad(p), nabla_grad(q))*dx
L2 = dot(nabla_grad(p_n), nabla_grad(q))*dx - (1/k)*div(u_)*q*dx

# Define variational problem for step 3
a3 = dot(u, v)*dx
L3 = dot(u_, v)*dx - k*dot(nabla_grad(p_ - p_n), v)*dx
```

Figure 7: Variational problems

For each step, we assemble the left-side of the system to not have to do it at each iteration in the loop.

```
# Assemble matrices
A1 = assemble(a1)
A2 = assemble(a2)
A3 = assemble(a3)
```

Figure 8: Assemble

And finally, for the number of steps we want, we solve a new velocity and a new pressure in a loop. We assemble the right-and side, and solve the equations.

```
# Time-stepping
t = 0
for n in range(num_steps):
    # Update current time
    t += dt

    # Step 1: Tentative velocity step
    b1 = assemble(L1)
    [bc.apply(b1) for bc in bcu]
    solve(A1, u_.vector(), b1, 'bicgstab', 'hypre_amg')

    # Step 2: Pressure correction step
    b2 = assemble(L2)
    [bc.apply(b2) for bc in bcp]
    solve(A2, p_.vector(), b2, 'bicgstab', 'hypre_amg')

    # Step 3: Velocity correction step
    b3 = assemble(L3)
    solve(A3, u_.vector(), b3, 'cg', 'sor')
```

Figure 9: Solving the equations

To plot the results, the best way is to save each velocity and pressure into a file. Then, we can view the whole animation using ParaView.
For that, we create two files during the initiation of the simulation, one for the velocity and one for the pressure.

```
# Create XDMF files for visualization output
xdmffile_u = XDMFFile('navier_stokes_cylinder/velocity1000_coef.xdmf')
xdmffile_p = XDMFFile('navier_stokes_cylinder/pressure1000_coef.xdmf')
```

Figure 10: Create a file to save our animation

And we save our results in our files for each iteration of the loop.

```
# Save solution to file (XDMF/HDF5)
xdmffile_u.write(u_, t)
xdmffile_p.write(p_, t)
```

Figure 11: Saving the results

5 Results

Using ParaView, we can open and visualize our .xdmf files containing our animation. In options, I represent the velocity with arrow scaled with its value. The pressure is represented by the color: more there are pressure, more it will extend to the red.

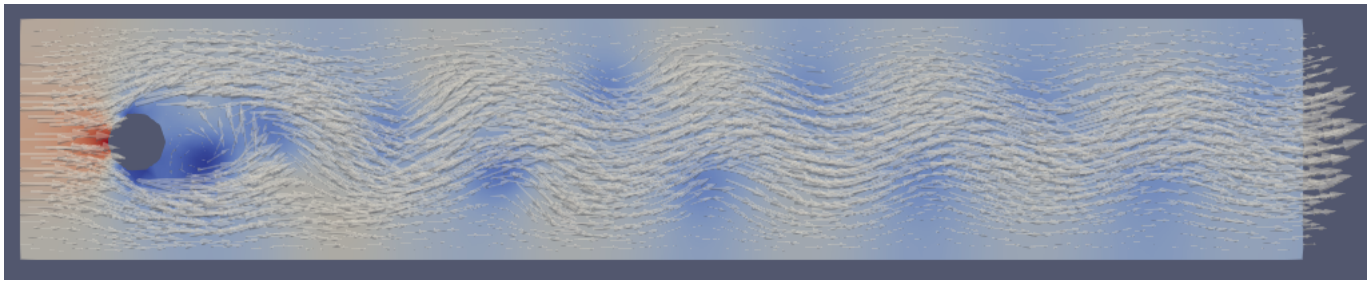


Figure 12: Flow around a cylinder

They are many points interesting in this visualization of the flow around a cylinder. Firstly, we can see the high pressure where the flow hits the cylinder. Then it divides and bypass the cylinder by both sides. We can notice the low pressure behind the cylinder. It's because the flow doesn't go there, except some low velocity area.

After bypassing the cylinder, the flow does an sinusoidal path. It's explained by the fact that the cylinder is not exactly in the center of the channel. Actually, the channel has an height of 0.41, and the center is the cylinder is at 0.2. The flow has more space above the cylinder. Both parts of the flow join behind the cylinder with not the same velocity and pressure. So the flow doesn't go straight forward and have a sinusoidal path.

Now let's see how reacts the flow in a narrow gap:

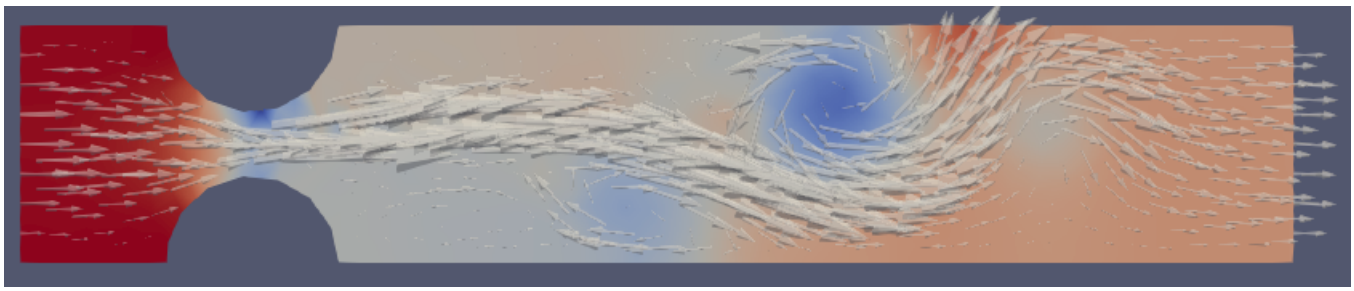


Figure 13: Flow in a narrow gap

The result of this simulation is as expected. The small space before the gap sees his pressure increasing. That results that in the gap the pressure is low and the velocity increase.

Thus, we have at the end of the gap a strong velocity. This flow doesn't go straight for the same reason as before. The gap is not centered compared of the channel. The flow will oscillate and create big holes of low pressure.

It's very interesting to see how just a small difference of space before the gap by no centering the gap can create big deformations on the flow and like small "anticyclone".
It represents very well the butterfly effect!

6 Conclusion

These simulations are very interesting and we can easily understand and visualize physic problems. It's also amazing how FEniCS is easy and practical to use, with some knowledge about PDE to formulate the variational problems. We showed here some simple simulations, but we can do more practical problems like described in the introduction, like simulating blood flow. There is also the three dimensions domain that we haven't shown here. Navier-Stokes equations in three dimensions are more interesting and complicated. Even their solutions have not been proven yet! It's one of the seven Millennium Prize problems in mathematics. Here is the formulation of the problem:

Prove or give a counter-example of the following statement:

In three space dimensions and time, given an initial velocity field, there exists a vector velocity and a scalar pressure field, which are both smooth and globally defined, that solve the Navier–Stokes equations.

Bibliography

- **Lecture notes by Achim Schroll**
- **FEniCS Project Tutorial:** https://fenicsproject.org/pub/tutorial/html/ftut1.html
- **Wikipedia:** https://en.wikipedia.org/wiki/Projection_method_(fluid_dynamics)
- **Chorin's article:** https://projecteuclid.org/download/pdf_1/euclid.bams/1183529112
- **Advanced Studies in Theoretical Physics:** http://www.m-hikari.com/astp/astp2018/astp1-4-2018/p/cardenasASTP1-4-2018-1.pdf
- **Wikipedia:** https://en.wikipedia.org/wiki/Navier%E2%80%93Stokes_existence_and_smoothness