A numerical analysis of the Navier Stokes Equations

Anders Hafreager

December 5, 2012

Abstract

A finite element method for the numerical solution of the incompressible Navier-Stokes equation (NSE) is derived. The method is called Incremental Pressure Correction Scheme (IPCS)[4], and is an extension of the method of Chorin in 1968[1]. We test the numerical stability of the method and compare the results to analytical solutions.

1 Introduction

The physics of fluids has been a major field of study and goes back to the days of ancient Greece [3] when the first formulations of fluid beaviour was written

> *"Any object, wholly or partially immersed in a fluid, is buoyed up by a force equal to the weight of the fluid displaced by the object."* - Archimedes of Syracuse.

In modern mathematical language, we write this as

$$F_{net} = m_b g - \rho V_f g.$$

After Bernoulli published *Hydrodynamica* (1738), where he introduced a mathematical framework, more complicated systems were available for theoretical study. A large amount of the problems we meet in vector calculus today are named after physicists after being first used on fluid mechanical problems. Lagrange, Euler, Laplace and Poisson are names that every student who touch differential equations probably have heard of. In 1822, Claude Louis Marie Henri Navier derived a set of equations that describe the velocity- and pressure field of a fluid with a term describing the viscosity, called the Navier Stokes equations.[5] These equations have been studied a lot since their first appearance, and are among the most important mathematical tools in modern engineering.
We start by deriving the NSE from conservation of momentum before we look at a few analytically solvable problems that we will use to validate the numerical implementation. The differential equation is then formulated as a variational problem so we can solve it with a finite element method. We will only study the incompressible equation where we assume that $\nabla \cdot u = 0$.

2 Derivation

There are many ways to derive the NSE. Most of them are equivalent because they simply assume conservation of mass and momentum. NSE describes the two most important properties in a fluid, namely the pressure field p and velocity field $\mathbf{u}$. We will use the notation of the material derivative

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbf{u} \cdot \nabla,$$

and the integral theorems known from vector calculus. In addition, we will work with second order tensors, dyads, which we won't discuss in detail. These expressions will be used directly in the computer program FEniCS[2], where the mathematics are handled.

2.1 Conservation laws

Consider a property L that is measureable within a finite, arbitrary volume Ω with boundary $\partial\Omega$. The rate of change equals the amount that is created/consumed inside the volume in addition to what flows through the boundary. This can be expressed as

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_\Omega L\mathrm{d}V = -\int_{\partial\Omega} L\mathbf{u}\cdot\mathbf{n}\mathrm{d}A - \int_\Omega Q\mathrm{d}V,$$

where $\mathbf{n}$ is the normal vector to the boundary pointing outwards, $\mathbf{u}$ is the fluid velocity and Q represents the sources or sinks inside Ω. If we apply the divergence theorem, this becomes

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_\Omega L\mathrm{d}V = -\int_\Omega \nabla\cdot(L\mathbf{u})\mathrm{d}V - \int_\Omega Q\mathrm{d}V,$$

or simply

$$\int_\Omega \left(\frac{\partial L}{\partial t} + \nabla\cdot(L\mathbf{u}) + Q\right)\mathrm{d}V = 0.$$

Since the volume Ω is arbitrary chosen, the integrand must be zero

$$\frac{\partial L}{\partial t} + \nabla\cdot(L\mathbf{u}) + Q = 0. \tag{1}$$

This is the differential form of conservation laws that easily can be applied to any scalar field. It can also be used for vector fields where $L\mathbf{u} \to \mathbf{L}\mathbf{u}$ is a dyadic tensor. This will be used to use the conservation law on the momentum $\rho\mathbf{u}$.

2.2 Conservation of mass

By assuming that no mass is created or destroyed, we apply the conservation equation on the density

$$\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\mathbf{u}) = 0.$$

If the fluid is incompressible, ρ is a constant, reducing the above equation to

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

which will play an important part of choice of finite element formulation later on.

2.3 Conservation of momentum

We apply the conservation law on the momentum per unit volume $\rho\mathbf{u}$

$$\frac{\partial\rho\mathbf{u}}{\partial t} + \nabla\cdot(\rho\mathbf{u}\mathbf{u}) + \mathbf{Q} = 0.$$

The $\mathbf{Q}$ term, the source or sink, is simply the body force, i.e. external forces that we denote by $\mathbf{b}$. Writing out the dyadic term gives

$$\mathbf{u}\frac{\partial\rho}{\partial t} + \rho\frac{\partial\mathbf{u}}{\partial t} + \mathbf{u}(\mathbf{u}\cdot\nabla\rho) + \rho\mathbf{u}(\nabla\cdot\mathbf{u}) + \rho(\mathbf{u}\cdot\nabla)\mathbf{u} = \mathbf{b},$$

and by rearranging

$$\begin{aligned}\mathbf{b} &= \mathbf{u}\Big(\frac{\partial\rho}{\partial t} + \mathbf{u}\cdot\nabla\rho + \rho\nabla\cdot\mathbf{u}\Big) + \rho\Big(\frac{\partial\mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}\Big)\\ &= \mathbf{u}\Big(\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\mathbf{u})\Big) + \rho\Big(\frac{\partial\mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}\Big),\end{aligned}$$

we recognize the first term as the mass conservation equation, which is equal to zero. Conservation of momentum simply reduces to

$$\mathbf{b} = \rho\Big(\frac{\partial\mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}\Big) = \rho\frac{D\mathbf{u}}{Dt},$$

where we have the material derivative of the fluid velocity multiplied with the density.

2.4 The body force

The source of momentum inside a volume is caused by body forces $\mathbf{b}$, but these can be divided into different kinds of forces; distant forces like gravity or electrostatic forces and a stress term (friction and normal forces, i.e. stress forces). By using the stress tensor, the body force can be written as

$$\mathbf{b} = \nabla \cdot \sigma + \mathbf{f},$$

which inserted in the momentum conservation looks like

$$\rho \frac{D\mathbf{u}}{Dt} = \nabla \cdot \sigma + \mathbf{f},$$

where σ is the stress tensor, and $\mathbf{f}$ is the sum of distant forces. The stress tensor is defined in the usual way

$$\begin{aligned} \sigma &= \begin{pmatrix} \sigma_x & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_y & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma z \end{pmatrix} \\ &\equiv -p\mathbb{1} + \mathbb{T}, \end{aligned}$$

where $\mathbb{1}$ is the identity matrix, p is the average pressure

$$p = -\frac{1}{3}(\sigma_x + \sigma_y + \sigma_z),$$

and $\mathbb{T}$ is the deviatoric stress tensor

$$\mathbb{T} = \begin{pmatrix} \sigma_x + p & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_y + p & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma z + p \end{pmatrix}.$$

The conservation equation now looks like

$$\rho \frac{D\mathbf{u}}{Dt} = -\nabla p + \nabla \cdot \mathbb{T} + \mathbf{f}.$$

This is the Navier Stokes equation in its most general form. There are many degrees of freedom since $\mathbb{T}$ can be defined in different ways depending on the fluid properties. A usual simplification is to assume that the divergence of $\mathbb{T}$ is proportional to $\nabla^2\mathbf{u}$ with proportionality constant μ

$$\nabla \cdot \mathbb{T} = \mu \nabla^2 \mathbf{u},$$

where μ is called viscosity. NSE then looks like

$$\rho \frac{D\mathbf{u}}{Dt} = -\nabla p + \mu \nabla^2 \mathbf{u} + \mathbf{f}, \tag{3}$$

and this is the equation we will analyze.

3 Analytical solutions

Different geometries give rise to very different solutions, and there are a few systems that hase a closed form solution. One system of interest is flow in a 2 dimensional canal where we apply a constant pressure gradient in the x-direction. Such flows are called *Hagen-Poiseuille flow*. This is a stationary solution where the time derivative $\partial_t \mathbf{u} = 0$. We can also look at *Starting Couette flow* where there is no pressure gradient, but one of the walls inside the canal suddenly starts to move with a velocity U.

3.1 2 dimensional Hagen-Poiseuille flow

We will now look at flow in a 2 dimensional box of length L (x-direction) and height d where we assume that $\mathbf{u} = (u_x, 0, 0)$, i.e. the only non zero component is in the x-direction. The boundaries are not moving, and we will use $u = u_x$ for simple notation. If we assume that $\partial_x u_x = 0$, the NSE looks like

$$\rho\frac{\partial u}{\partial t} = -\frac{\partial p}{\partial x} + \mu\frac{\partial^2 u}{\partial y^2} = -\beta + \mu\frac{\partial^2 u}{\partial y^2}$$

where $\beta = \partial_x p$. If we also assume stationary flow, this reduces to

$$\frac{\partial^2 u}{\partial y^2} = \frac{\beta}{\mu}$$

which has the solution

$$u(y) = \frac{\beta}{2\mu}y(d-y).$$

The maximum velocity is found in the middle of the box $y = d/2$

$$u_{max} = u(d/2) = \frac{\beta}{2\mu}\frac{d^2}{4}. \tag{4}$$

The pressure p is linear, so the pressure gradient is a constant

$$\beta = \frac{P_A - P_B}{L},$$

which inserted in (4) gives

$$u_{max} = \frac{(P_A - P_B)}{8\mu L}d^2.$$

3.2 Starting Couette flow

4 Numerical solution of the incompressible equation

In order to solve this with a finite element method, we need to discretize the equation both in the time dimension and the spatial dimension. The only time dependent part of the equation is in the material derivative. The spatial discretization will be taken care of by the FEniCS program, so we only need to work on the time derivative. We rewrite NSE to

$$\frac{\partial \mathbf{u}}{\partial t} = \nu\nabla^2\mathbf{u} - \frac{1}{\rho}\nabla p + \mathbf{f} - (\mathbf{u}\cdot\nabla)\mathbf{u},$$

where we have defined the kinematic viscosity $\nu = \mu/\rho$, defined $\mathbf{F} = \mathbf{f}/\rho$, force per unit volume. We will use a scheme where we can exploit the fact that $\nabla\cdot\mathbf{u} = 0$ and manage to calculate p^{n+1}.

4.1 The classical splitting method

We use the Forward Euler method to find an expression for $\mathbf{u}^{n+1}$, but we evaluate the pressure also at the time step $n+1$. This will give an extra degree of freedom that we can use to make sure that $\mathbf{u}$ behaves as we know it should

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t\nu\nabla^2\mathbf{u}^n - \frac{\Delta t}{\rho}\nabla p^{n+1} + \Delta t\mathbf{f} - \Delta t(\mathbf{u}^n\cdot\nabla)\mathbf{u}^n. \tag{5}$$

It is now possible to choose ∇p^{n+1} so that $\nabla\cdot\mathbf{u}^{n+1} = 0$. By using a temporary $\mathbf{u}^*$ which we define as

$$\mathbf{u}^* = \mathbf{u}^n + \Delta t\nu\nabla^2\mathbf{u}^n - \beta\frac{\Delta t}{\rho}\nabla p^n + \Delta t\mathbf{f} - \Delta t(\mathbf{u}^n\cdot\nabla)\mathbf{u}^n, \tag{6}$$

note the factor β in the pressure gradient term. Think of this as an intermediate solution. We now choose a $\delta\mathbf{u}$ so that

$$\begin{aligned}\mathbf{u}^{n+1} &= \mathbf{u}^* + \delta\mathbf{u} \\ \delta\mathbf{u} &= \mathbf{u}^{n+1} - \mathbf{u}^* \\ &= -\frac{\Delta t}{\rho}\nabla\Phi,\end{aligned}$$

by subtracting (6) from (5), where $\Phi = p^{n-1} - \beta p^n$. The incompressibility constraint can now be written as

$$\nabla \cdot \mathbf{u}^{n+1} = \nabla \cdot (\mathbf{u}^* + \delta\mathbf{u}) = 0,$$

or

$$\nabla \cdot \mathbf{u}^* = -\nabla \cdot \delta\mathbf{u}.$$

We apply this on our expression for $\delta\mathbf{u}$

$$\nabla^2\Phi = \frac{\rho}{\Delta t}\nabla \cdot \mathbf{u}^*,$$

and recognize this as a Poisson equation for Φ that is easily solved since we know $\mathbf{u}^*$. We now have everything we need to evolve the system in time where the velocity is given as

$$\mathbf{u}^{n+1} = \mathbf{u}^* - \frac{\Delta t}{\rho}\nabla\Phi, \tag{7}$$

and the pressure

$$\mathbf{p}^{n+1} = \Phi + \beta p^n. \tag{8}$$

5 A Finite Element Method

Assume now that we have given the initial pressure field p and the initial velocity field $\mathbf{u}_0$. Following the recipe described above, we start by finding the $\mathbf{u}^*$.

5.1 Variational form for $\mathbf{u}^*$

We introduce basis vectors living in the test space $V^{(u)}$, and seek $\mathbf{u}^*, \mathbf{u}^{n+1} \in V^{(u)}$. We create a variational form of (6) by multiplying by a test vector $v \in V^{(u)}$ and integrate

$$\begin{aligned}\frac{1}{\Delta t}\int_\Omega (\mathbf{u}^* - \mathbf{u}^n)\cdot\mathbf{v}^{(u)}\mathrm{d}\Omega - \nu\int_\Omega (\nabla^2\mathbf{u}^n)\cdot\mathbf{v}^{(u)}\mathrm{d}\Omega \\ + \frac{\beta}{\rho}\int_\Omega \nabla p^n\cdot\mathbf{v}^{(u)}\mathrm{d}\Omega - \int_\Omega \mathbf{f}\cdot\mathbf{v}^{(u)}\mathrm{d}\Omega + \int_\Omega (\mathbf{u}^n\cdot\nabla)\mathbf{u}^n\cdot\mathbf{v}^{(u)}\mathrm{d}\Omega = 0.\end{aligned}$$

By integrating the Laplace term and the pressure term by parts, we get

$$\begin{aligned}\frac{1}{\Delta t}\int_\Omega (\mathbf{u}^* - \mathbf{u}^n)\cdot\mathbf{v}^{(u)}\mathrm{d}\Omega + \nu\int_\Omega \nabla\mathbf{u}^n\cdot\nabla\mathbf{v}^{(u)}\mathrm{d}\Omega \\ - \frac{\beta}{\rho}\int_\Omega p^n\nabla\cdot\mathbf{v}^{(u)}\mathrm{d}\Omega - \int_\Omega \mathbf{f}\cdot\mathbf{v}^{(u)}\mathrm{d}\Omega + \int_\Omega (\mathbf{u}^n\cdot\nabla)\mathbf{u}^n\cdot\mathbf{v}^{(u)}\mathrm{d}\Omega \\ = \int_{\partial\Omega_{N,u}} (\nu\frac{\partial\mathbf{u}}{\partial n} - p^n\mathbf{n})\cdot\mathbf{v}^{(u)}\mathrm{d}s\end{aligned}$$

5.2 Variational form for Φ

To solve the Poisson equation for Φ, we multiply by a test function $v^{(\Phi)} \in V^{(\Phi)}$ and integrate

$$\int_\Omega \nabla^2 \Phi v^{(\Phi)} \mathrm{d}\Omega = \frac{\rho}{\Delta t} \int_\Omega \nabla \cdot \mathbf{u}^* v^{(\Phi)} \mathrm{d}\Omega$$
$$\int_\Omega \nabla \Phi \cdot \nabla v^{(\Phi)} \mathrm{d}\Omega = -\frac{\rho}{\Delta t} \int_\Omega \nabla \cdot \mathbf{u}^* v^{(\Phi)} \mathrm{d}\Omega + \int_{\partial\Omega_{n,\Phi}} \frac{\partial \Phi}{\partial n} v^{(\Phi)} \mathrm{d}s$$

When we have found Φ and $\mathbf{u}^*$, we can use a variational form of (7)

$$\int_\Omega \mathbf{u}^{n+1} \cdot \mathbf{v}^{(u)} \mathrm{d}\Omega = \int_\Omega (\mathbf{u}^* - \frac{\Delta t}{\rho} \nabla \Phi) \cdot \mathbf{v}^{(u)} \mathrm{d}\Omega,$$

similarly for the pressure

$$\int_\Omega p^{n+1} v^{(\Phi)} \mathrm{d}\Omega = \int_\Omega (\Phi + \beta p^n) v^{(\Phi)} \mathrm{d}\Omega.$$

6 Numerical implementation

6.1 Stability criterion

The numerical solution is based on a discretized time derivative, and the stability criterion is given by [4]

$$\Delta t \leq \frac{h^2}{2\nu + Uh},$$

where h is the minimum element size and U is a characteristic size of the velocity. For water and air, the viscosity is less than 10^{-4}, so the stability criterion goes like

$$\Delta t \leq \frac{h}{U},$$

which we will study in more detail later in this report.

7 Verification

7.1 2 dimensional Hagen-Poiseuille flow

Our adjustable parameters are

- Δt - time step
- ΔP - pressure difference
- μ - viscosity
- L - distance between A and B in pressure difference
- d - box height

We use a uniform mesh, a rectangle, of size $L \times d$ with 10 elements in each direction. A FEniCS implementation of the Hagen-Poiseuille flow can be found in Appendix A. We let FEniCS run until we reach a steady and compare the maximum thermal velocity to the theoretical max given by (4). The results can be found in table 1. Note that the different parameters carry units, but we will for simplicity only look at the numerical values.

#	Δp	μ	L	h	Analytical	FEniCS	Error
1	1	1	1	1	0.125	0.125	0.000
2	2	1	1	1	0.250	0.250	0.000
3	1	1	1	2	0.500	0.500	0.000

Table 1: Comparison of the analytical and the numerical solution of the pressure driven stationary system.

8 Appendices

A Poiseuilles law in FEniCS

```
from dolfin import *
import numpy as np

#mesh = UnitSquare(10,10)
mesh = Rectangle(0,0,2,1,20,20,'left')

# Define function spaces (P2-P1)
V = VectorFunctionSpace(mesh, "CG", 2)
Q = FunctionSpace(mesh, "CG", 1)

# Define trial and test functions
u = TrialFunction(V) # Trial function for u
p = TrialFunction(Q) # Trial function for p

v = TestFunction(V) # Test function for u
q = TestFunction(Q) # Test function for p

# Normal vector
n = FacetNormal(mesh)

# Set parameter values
dt = 0.01 # Timestep
T = 40      # Final time
nu = 0.5 # Kinematic viscosity
rho = 1.0     # Density

p_in = Constant(1.0) # Pressure at x=0
#p_in = Expression("sin(2*pi*t)", t=0.0)
p_out = Constant(0.0) # Pressure at x=1
# No slip makes sure that the velocity field is zero at the boundaries
noslip_1  = DirichletBC(V, (0, 0),
                        "on_boundary && \
                         x[1] < DOLFIN_EPS")

noslip_2  = DirichletBC(V, (0, 0),
                        "on_boundary && \
                         x[1] > 1 - DOLFIN_EPS")

# We apply a pressure at x=0, no pressure at
inflow  = DirichletBC(Q, p_in, "x[0] < DOLFIN_EPS")
outflow = DirichletBC(Q, p_out, "x[0] > 2.0 - DOLFIN_EPS")

bcu = [noslip_1, noslip_2]
bcp = [inflow, outflow]

# Create functions
u0 = Function(V)
u1 = Function(V)
p0 = Function(Q)
p1 = Function(Q)

# Define coefficients
k = Constant(dt)
f = Constant((0, 0))

# Tentative velocity step
U = 0.5*(u0 + u)

F1 = (1/k)*inner(u - u0, v)*dx \
    + nu*inner(grad(U), grad(v))*dx \
    - p0*div(v)*dx\
```

```
    - inner(f, v)*dx\
    + inner(grad(u0)*u0, v)*dx \
    + inner(p0*n,v)*ds\

a1 = lhs(F1)
L1 = rhs(F1)

# Calculation of phi
a2 = inner(grad(p), grad(q))*dx
L2 = inner(grad(p0),grad(q)) *dx\
     -(1.0/k)*div(u1)*q*dx

# Velocity update
a3 = inner(u, v)*dx
L3 = inner(u1, v)*dx - k*inner(grad(p1 - p0), v)*dx

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

# Create files for storing solution
ufile = File("results/velocity.pvd")
pfile = File("results/pressure.pvd")

# Time-stepping
t = dt
while t < T + DOLFIN_EPS:
    # Update pressure boundary condition
    p_in.t = t

    # Compute tentative velocity step
    b1 = assemble(L1)
    [bc.apply(A1, b1) for bc in bcu]
    solve(A1, u1.vector(), b1, "gmres", "default")

    # Pressure correction
    b2 = assemble(L2)
    [bc.apply(A2, b2) for bc in bcp]
    solve(A2, p1.vector(), b2, "gmres", "amg")

    # Velocity correction
    b3 = assemble(L3)
    [bc.apply(A3, b3) for bc in bcu]
    solve(A3, u1.vector(), b3, "gmres", "default")

    plot(u1, title="Velocity", rescale=True)

    # Save to file
    #ufile << u1
    #pfile << p1

    # Move to next time step
    u0.assign(u1)
    p0.assign(p1)
    t += dt

    u_vec = u1.vector()
    u_array = u_vec.array()
    print "t =", t
    print "max: ",u_array.max()


# Hold plot
interactive()
```

References

[1] Alexandre Joel Chorin. "Numerical Solution of the Navier-Stokes Equations*". In: *Math. Comp.* 22 (1968), pp. 745–762. DOI: 10.1090/S0025-5718-1968-0242392-2.

[2] *FEniCS Project*. URL: http://fenicsproject.org/.

[3] *Fluid Mechanics*.

[4] Prof. Hans Petter Langtangen. *Numerical methods for the Navier-Stokes equations*. URL: http://hplgit.github.com/INF5620/doc/notes/main_ns.pdf.

[5] *Navier-Stokes equations*. URL: http://www.cfd-online.com/Wiki/Navier-Stokes_equations#History.