Mek9250 - Mandatory Exercise

Florian Arbes

February 2021

Exercise 1

The following equation is given on the domain $\Omega = (0,1)^2$:

$$-\mu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)+\frac{\partial u}{\partial x}=0 \quad in \quad \Omega, \tag{1}$$

$$u = 0 \quad for \quad x = 0 \tag{2}$$

$$u = 1 \quad for \quad x = 1 \tag{3}$$

$$\frac{\partial u}{\partial n} = 0 \quad for \quad y = 0 \quad and \quad y = 1 \tag{4}$$

a) An analytical solution

Ansatz: $u(x,y) = u(x)$, which means $\frac{\partial u}{\partial y} = 0$ and $\frac{\partial^2 u}{\partial y^2} = 0$. The PDE then simplifies to:

$$\frac{\partial u}{\partial x} = \mu\left(\frac{\partial^2 u}{\partial x^2} + 0\right)$$

This ODE can be solved easily:

$$\int_\Omega \frac{\partial u}{\partial x}dx = \mu\int_\Omega \frac{\partial^2 u}{\partial x^2}dx$$

$$u = \mu\frac{\partial u}{\partial x}dx + C$$

Thus, $u(x)$ has the form $u(x) = Ae^{Bx} + C$, with the derivatives $\frac{\partial u}{\partial x} = ABe^{Bx}$ and $\frac{\partial^2 u}{\partial x^2} = AB^2e^{Bx}$. This can be plugged into the ODE:

$$ABe^{Bx} = \mu\left(AB^2e^{Bx} + 0\right) \Leftrightarrow B = \frac{1}{\mu}$$

If A and C can be chosen in a way, that they fulfill the boundary conditions, $u(x)$ is a solution to the PDE as well.

$$u = 0 \quad for \quad x = 0 \qquad \Rightarrow u(x=0) = Ae^{0B} + C = 0 \tag{5}$$
$$u = 1 \quad for \quad x = 1 \qquad \Rightarrow u(x=1) = Ae^{1B} + C = 1 \tag{6}$$
$$\frac{\partial u}{\partial n} = 0 \quad for \quad y = 0 \quad and \quad y = 1 \quad \Rightarrow \frac{\partial u}{\partial y} = 0 \quad (for \quad all \quad y \quad true) \tag{7}$$

Subtracting (5) from (6) leads to:

$$Ae^{1B} - A = 1 \Leftrightarrow A = \frac{1}{e^B - 1}$$

From (5) we get

$$C = -A = -\frac{1}{e^B - 1}$$

The solution therefore is:

$$u(x,y) = \frac{1}{e^B - 1} e^{Bx} - \frac{1}{e^B - 1} = (e^{Bx} - 1)\frac{1}{e^B - 1} = \frac{e^{\frac{x}{\mu}} - 1}{e^{\frac{1}{\mu}} - 1}$$

b) Numerical error for various h and various μ

Figure 1 shows the analytical and numerical solution for various mu (h = 1/8). As h decreased, the error decreases. The error can be estimated with

$$||u - u_h||_1 \leq C_\alpha h^\alpha$$

and

$$||u - u_h||_0 \leq C_\beta h^\beta$$

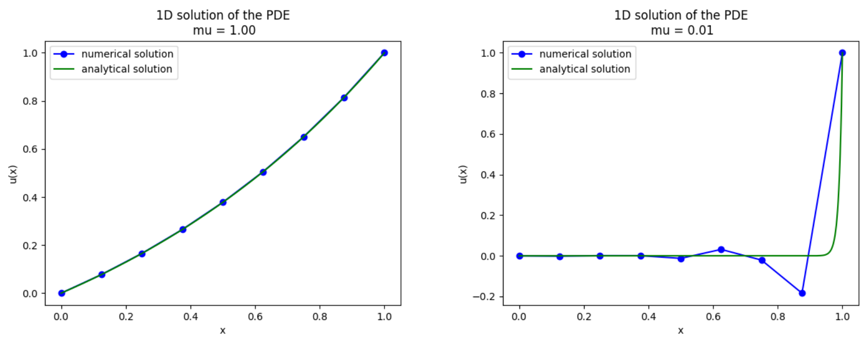
A curve fit is used to approximate C_α, α, C_β and β (see Figure 2). Since the scheme is $\mathcal{O}(h^2)$, α and β should equal 2. The numerical approximations are listed in Table 1 for different values of μ.

mu	C_{alpha}	alpha	C_{beta}	beta
1.00	0.1230	1.9538	0.0807	1.9990
0.30	1.2157	1.8635	0.2559	1.9765
0.10	8.3900	1.6440	0.9624	1.8894

b) Numerical error with SUPG stabilization

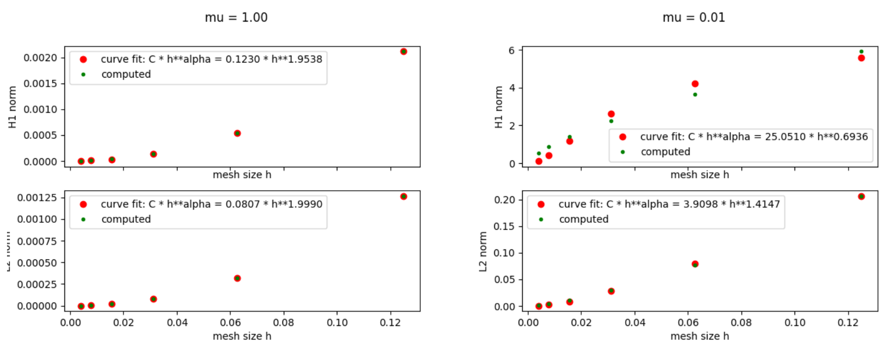
After introducing a stabilization, the numerical results look slightly different (see Figure 3). The error estimates are given in Table 2

mu	C_{alpha}	alpha	C_{beta}	beta
1.00	0.1220	1.9519	0.0845	2.0117
0.30	0.9965	1.8073	0.3066	2.0286
0.10	5.8300	1.5390	0.9880	1.8970



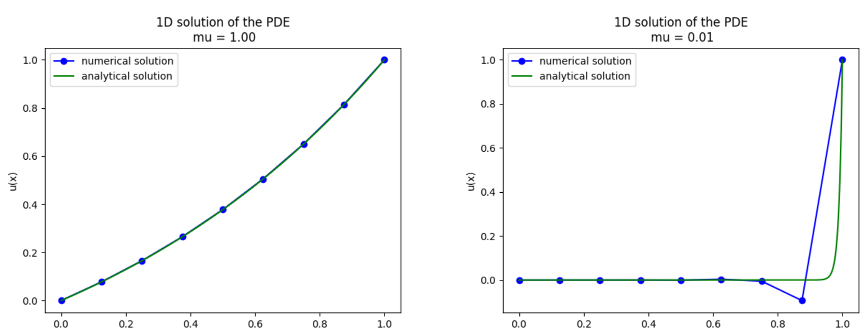
(a) mu = 1.0 (b) mu = 0.01

Figure 1: Numerical solutions (h=1/8).



(a) mu = 1.0 (b) mu = 0.01

Figure 2: curve fit for various h.



(a) mu = 1.0 (b) mu = 0.01

Figure 3: Numerical solutions with SUPG stabilization (h=1/8).

Exercise 2

The Navier-Stokes equations for incompressible fluids ($\nabla \cdot u = 0$) is defined as followed:

$$\varrho \frac{\partial u}{\partial t} + \varrho(u \cdot \nabla)u = \mu \nabla^2 u - \nabla \cdot pI + \varrho g \tag{8}$$

The terms from left to right are the acceleration, convection, diffusion, pressure gradient and the body forces (such as gravity). The terms can be separated like that:

$$\varrho \frac{\partial u}{\partial t} = F(u, p)$$

The equations can be discretized in time, where u^{n+1} denotes the time-step that needs to be computed. u^n is then the known time-step. A fully explicit time discretization would be a forward euler scheme:

$$\varrho \frac{u^{n+1} - u^n}{\Delta t} = F(u^n, p^n)$$

a) Implement a solver for the benchmark problem.

See code. It code can be executed from the command line like that:

```
python chorin_proj_cylinder.py --d_velocity 2 --d_pressure 1 --explicit 1
```

A semi-implicit discretization can be executed from the command line like that:

```
python chorin_proj_cylinder.py --d_velocity 2 --d_pressure 1 --explicit 0
```

b) Stability requirement

The cfl number was found to be around 0.05 in order to have a stable scheme

c) Drag and lift

The drag and lift coefficients were found to be 3.1846 and 0.7358 respectively, which is in the given bounds (see also Figure 4).

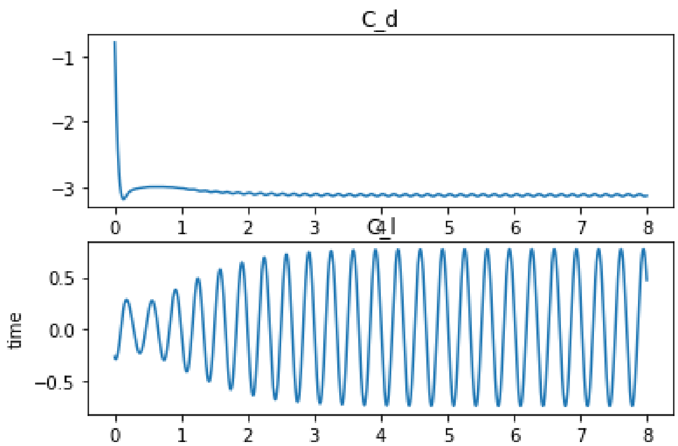


Figure 4: Drag and lift coefficients