TECHNISCHE UNIVERSITÄT WIEN
Vienna | Austria

INSTITUTE FOR ANALYSIS AND SCIENTIFIC COMPUTING
WIEDNER HAUPTSTRASSE 8-10
1040 WIEN, AUSTRIA

DR.
PHILIP LEDERER
email: philip.lederer@tuwien.ac.at

Finite Element Methods in Computational Fluid Dynamics

Exercise 3 – January 2023

Example 3.1

Solve the convection-diffusion problem

$$-\varepsilon\Delta u + b \cdot \nabla u = f \tag{1}$$

with homogeneous Dirichlet boundary conditions on $\Omega = (0,1)$ where $b = (b_1, b_2)$ and the right hand side

$$f = b_1\Big(y - \frac{e^{b_2 y/\varepsilon} - 1}{e^{b_2/\varepsilon} - 1}\Big) + b_2\Big(x - \frac{e^{b_1 x/\varepsilon} - 1}{e^{b_1/\varepsilon} - 1}\Big).$$

For $\varepsilon > 0$ the exact solution is given by

$$u = \Big(x - \frac{e^{b_1 x/\varepsilon} - 1}{e^{b_1/\varepsilon} - 1}\Big)\Big(y - \frac{e^{b_2 y/\varepsilon} - 1}{e^{b_2/\varepsilon} - 1}\Big). \tag{2}$$

Use a standard conforming finite element method of order $k = 1,2,3$ and compare the L^2-norm error for the case $b = (2,1)$ and $\varepsilon = 0.01$ and levels of refinemend $L = 0,1,2,3$ given by the unstructured mesh produced by the function

```
import ngsolve.meshes as ngm

def GetMesh(L = 0):
    mesh = ngm.MakeStructured2DMesh(quads = False, nx = 2,ny = 2)
    for i in range(L+1):
        mesh.Refine()
    return mesh

# for example
# mesh = GetMesh(L = 0)
```

Example 3.2

Implement the streamline diffusion stabilization with the stabilization parameter $\alpha = h/|b|$ (you can use the `meshsize` function of NGSolve for h) when convection dominates $\mathcal{P}_h \geq 1$ and recalculate the errors as in the first example.

Example 3.3

Implement the upwind HDG method with interior penalty parameter $\alpha = 3$ and order $k = 1,2,3$. To define the upwind value use

```
b = ...
n = specialcf.normal(2)
uup = IfPos(b * n, u, uhat).
```

Recalculate the errors as in the first example. Do not forget to "glue" the element values to the facet variables on the outflow boundary.

Example 3.4

Solve the lid driven cavity problem with the $P2$-bubble element: Find u, p such that

$$
\begin{aligned}
-\nu\Delta u + u \cdot \nabla u + \nabla p &= 0 && \text{in } \Omega,\\
u &= 0 && \text{on } \partial\Omega \setminus \Gamma_{top},\\
u &= (u_1, 0) && \text{on } \Gamma_{top},
\end{aligned}
$$

where $\Omega = (0,1)^2$, $\Gamma_{top} = [0,1] \times \{1\}$ and

$$
u_1(x) = \begin{cases} 1 - \frac{1}{4}(1 - \cos(\frac{x_1 - x}{x_1}\pi))^2 & \text{for } x \in [0, x_1],\\ 1 & \text{for } x \in (x_1, 1 - x_1),\\ 1 - \frac{1}{4}(1 - \cos(\frac{x - (1 - x_1)}{x_1}\pi))^2 & \text{for } x \in [1 - x_1, 1], \end{cases}
$$

with $x_1 = 0.1$. Use the Newton and the Picard iterative method to solve the problem on the mesh from the first example with the levels $L = 0, 1, 2, 3$ and the viscosity $\nu = 0.01, 0.002, 0.001$. Use the skew-symmetric version for the convection. Present a table with the number of iterations of the methods for the different levels and different viscosities. You can stop the iteration if $\sqrt{r_h^k \cdot \delta\underline{U}^{k+1}} \leq 10^{-10}$, where $\delta\underline{U}^{k+1} = (\underline{u}^{k+1}, \underline{p}^{k+1}) - (\underline{u}^k, \underline{p}^k)$ and $r_h^k = (r_{u,h}^k, r_{p,h}^k)$.

Hints:

- You can use the `IfPos` coefficient function of NGSolve to implement u_1.
- To evaluate the residual including the term $c(u_h^k, u_h^k, v_h)$ use the `Apply` function of a BilinearForm

```
a = Bilinearform(...)
a += ... # include nonlinear term here
gfu = GridFunction(...)
r1 = gfu.vec.CreateVector()

# returns the vector given by a substitution
# of the TrialFunction u by gfu in the blf
a.Apply(gfu.vec, r1)
```

Example 3.5

Solve the benchmark problem "DFG flow around cylinder benchmark 2D-2, time-periodic case Re=100" that can be found at `http://www.featflow.de/en/benchmarks/cfdbenchmarking/flow/dfg_benchmark2_re100.html`. Use an arbitrary inf-sup stable finite element method (for example the Taylor-Hood element) and implement the θ-scheme for arbitrary values $\theta \in [0,1]$. Calculate the drag and lift coefficients given by

$$
\begin{aligned}
c_d &= 20 \int_{\Gamma_\circ} \nu[((\nabla u)n) \cdot \tau]n[1] - pn[0] \,\mathrm{ds}\\
c_l &= -20 \int_{\Gamma_\circ} \nu[((\nabla u)n) \cdot \tau]n[0] - pn[1] \,\mathrm{ds}
\end{aligned}
$$

where n and $\tau = (n[1], -n[0])$ are the normal and tangential vector respectively and $\Gamma_\circ$ is the boundary of the obstacle within the time interval $t \in [0,5]$. To evaluate L^2 functions (such as the gradient of your approximation) use the NGSolve function

```
bndgrad =  BoundaryFromVolumeCF(...)
```

Compare your results for different time steps and different θ with the values presented on the benchmark homepage (you can download them there). As initial velocity you can use a Stokes solution. To solve the nonlinear systems you can use the Newton and Picard solver from the previous examples. For this first derive the linearized sytem for the update $(\delta u, \delta p)$ and modify you solver appropriately.

Hints:

1. You can use the `navierstokes.py` tutorial of NGSolve as starting point.
2. Your linearized problem of the update now includes the mass matrix.
3. If your forces are not accurate enough try to decrease the meshsize at $\Gamma_\circ$ and increase the order of approximation. Always curve the mesh with the same order as you use for your approximation.