Numerische Methoden in der Strömungsmechanik
OvGU WiSe 2023/24

Practical Exercises 5

In this exercise, we study stationary convection-diffusion equations and different stabilization techniques.

Problem 5.1 (Changing the equation)
We consider the following convection-diffusion equation

$$\begin{aligned} -\epsilon_0 \Delta u + \boldsymbol{\beta} \cdot \nabla u = 0 \quad & \text{in } \Omega = (0,1)^2, \\ u = g \quad & \text{on } \partial\Omega, \end{aligned}$$

with diffusion coefficient $\epsilon_0 > 0$ and convective field $\boldsymbol{\beta} = (1,1)^t$.

First, implement the correct Dirichlet data for this problem. We consider the following discontinuous function

$$g(x,y) = \begin{cases} 1 & \text{if } y \leq 0.5, \\ 0 & \text{else.} \end{cases}$$

which you have to implement in the file `problem.h`.

The weak formulation of the convection-diffusion equation is given by $a(u,\phi) = (f,\phi)$ for all $\phi \in V_h$, where

$$a(u,\phi) = \epsilon_0(\nabla u, \nabla\phi) + (\beta \cdot \nabla u, \phi).$$

The variational formulation is implemented in the function

```
1 Form(VectorIterator b, const FemFunction& U, const TestFunction& N)
```

The parameters `FemFunction& U` and `TestFunction& N` are the solution u and the test function ϕ. You can access the value of the testfunction and its derivative by $\phi(x,y) \sim$ `N.m()`, $\partial_x\phi(x,y) \sim$ `N.x()` and $\partial_y\phi(x,y) \sim$ `N.y()`. The solution `U` itself is a vector so that it holds $u(x,y) \sim$ `U[0].m()` and so on. (The values `U[1], U[2]` are used for systems of differential equations and will be explained later.)

The form of the Laplace problem, i.e. the bilinear form $a(u,\phi) = (\nabla u, \nabla\phi)$ is already implemented in the file `myequation.cc`:

```
1 void MyEquation::Form(VectorIterator b, const FemFunction& U,
2     const TestFunction& N) const {
3         b[0] += U[0].x()*N.x()+U[0].y()*N.y();
4 }
```

Analytical Jacobian (Finite difference version at the end of exercise)

As GASCOIGNE is designed to work directly with nonlinear problems, we have to implement above equation by providing both, the form $a(u, \phi)$, which is a semilinear mapping $a\colon V \times V \to \mathbb{R}$, and its (Gâteaux-) derivative $a'_u(u, \phi, \delta u)\colon V \times V \times V \to \mathbb{R}$,

$$a'_u(u, \phi, \delta u) := \lim_{s \to 0} \frac{a(u + s\,\delta u, \phi) - a(u, \phi)}{s}.$$

For the discrete space V_h and for fixed u the derivative can be interpreted as a matrix. If a is linear in its first component there holds $a'_u(u, \phi, \delta u) = a(\delta u, \phi)$. Form and matrix for the Laplace problem, i.e. the bilinear form $a(u, \phi) = (\nabla u, \nabla \phi)$ are already implemented in the file `myequation.cc`:

```
void MyEquation::Matrix(EntryMatrix& A, const FemFunction& U,
const TestFunction& M, const TestFunction& N) const {
A(0,0) += M.x()*N.x()+M.y()*N.y();
}
```

Here, `N` represents the test function ϕ, `M` the direction δu and `U[0]` the variable u. The x- and y-derivatives of the functions can be accessed through the methods `x()` and `y()`, their value through `m()`.

Task: Now please add the transport term given by $(\beta \cdot \nabla u, \phi)$ to `Form` and corresponding term `Matrix`.

The constructor of `MyEquation` copies `MyData` class

```
MyEquation(const MyData &PD) : data(PD) { }
```

Check `MyData` and make sure it reads ϵ_0 from the parameter file and solve the problem for $\epsilon_0 \in \{10^{-1}, 10^{-2}, 10^{-3}\}$ on meshes with a different refinement level. What behavior of the discrete solution do you observe? In the code ϵ_0 is called `visc`.

Problem 5.2 (Artificial Diffusion)

In order to avoid the unphysical oscillations of the discrete solution on coarse meshes one can add certain stabilization terms to the Galerkin formulation. At first, we add full artificial diffusion to the discrete equation. This means that the diffusion coefficient becomes mesh-dependent and is set to $\epsilon = \max\{\epsilon_0, h\}$. To implement this in GASCOIGNE you will need the function `point` in the `MyEquation` class:

```
void MyEquation::point(double h, const FemFunction& U,
                       const Vertex2d& v) const
```

This function is called before every call of `Form`. It contains parameters such as the local mesh-size h and the coordinates of the node v. Since the function `point` is defined as `const` it cannot change any variables of the class. Thus, every class variable that shall be modified in the `point`-function has to be defined as `mutable`, e.g.

```
1 mutable double 'variablename'
```

in the header file `equation.h`.

Solve the problem with $\epsilon_0 = 10^{-3}$ and for a series of mesh refinements. What is the effect on the solution compared to Problem 5.1?

Problem 5.3 (SUPG)

Now, we make use of the *Streamline Upwind/Petrov-Galerkin* (SUPG) method. Therefore, eliminate the artificial diffusion by setting the parameter ϵ back to its original value ϵ_0. Here, also artificial diffusion is added, but only in streamline direction. The stabilization term takes the following form:

$$S_h(u, \phi) = \delta(\boldsymbol{\beta} \cdot \nabla u, \boldsymbol{\beta} \cdot \nabla \phi).$$

Here, ϕ denotes a test function and δ is the stabilization parameter which is chosen to be cell-wise constant:

$$\delta = \delta_0 \left(\frac{\epsilon_0}{h^2} + \frac{\|\boldsymbol{\beta}\|_\infty}{h} \right)^{-1}, \quad \delta_0 > 0.$$

Implement this method in GASCOIGNE by modifying `Form` and `point`. Read the parameter δ_0 from the parameter file.

Solve the problem for $\epsilon_0 = 10^{-3}$ and $\delta_0 \in \{10^{-1}, 1, 10\}$. What is the effect on the solution? Compare your observations with the results of Problem 5.2. Do you still observe oscillations?

Problem 5.4 (Extra)

Finally, we consider another stabilization method, namely the *shock-capturing* method. Here, we still use SUPG stabilization, but add further artificial diffusion in the direction where oscillations in the SUPG method are observed. This non-linear stabilization term is defined as

$$S_h^{sc}(u,\phi) = \delta^{sc}(c(u)\nabla u, \nabla\phi)$$

with

$$c(u) = \begin{cases} 0 & \text{if } \nabla u = 0, \\ \frac{(\boldsymbol{\beta}\cdot\nabla u)^2}{\|\nabla u\|^2} & \text{else} \end{cases}$$

and $\delta^{sc} = \delta_0^{sc}\left(\frac{\epsilon_0}{h^2} + \frac{\|\boldsymbol{\beta}\|_\infty}{h}\right)^{-1}$.

Implement this method in GASCOIGNE by modifying `Form` and `point`. Solve this problem for $\epsilon_0 = 10^{-3}$ and $\delta_0^{sc} \in \{0.1, 0.5, 1.0\}$. What is the effect on the solution?

Hint: You may neglect the u-dependence of $c(u)$ in the Jacobian matrix.