Finite Elements (WS 2021) **Exercise 2**

Prof. Dr. Peter Bastian, Michal Tóth Submission date: 8 Nov 2021

IWR, Universität Heidelberg

Exercise 1 *Updating dune-npde*

As we update the *dune-npde* module during the semester, you need to get the current state before starting to solve a new programming exercise:

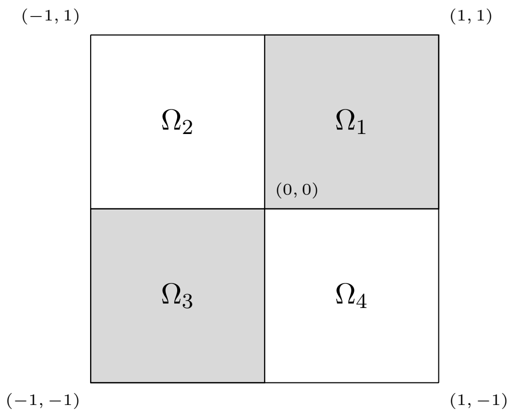
- Navigate to your *dune-npde* directory in a terminal
- Execute the commands

```
git stash
git pull
git stash pop
```

These *git* commands temporarily move your local changes to the stash, download the updates and apply your changes to the new version again.

(0 Points)

Exercise 2 *Analytical solution of heterogeneous heat equation*



On a bounded two-dimensional domain (see picture above) the equation describing stationary heat transfer should be solved:

$$\nabla \cdot (-\lambda \nabla u) = 0, \qquad \forall x \in \Omega, \text{ mit } \Omega = \bigcup_{i=1,\dots,4} \Omega_i,$$

λ is a piecewise constant given by

$$\lambda = \begin{cases} \lambda_1 & x \in \Omega_1 \cup \Omega_3 \\ \lambda_2 & x \in \Omega_2 \cup \Omega_4 \end{cases}.$$

1. Prove, that the following function in polar coordinates

$$p_i(r,\theta) = r^\delta (a_i \sin(\delta\theta) + b_i \cos(\delta\theta))$$

with constant coefficients $a_i, b_i, \delta \in \mathbb{R}$ in $\Omega \backslash (0,0)$ is harmonical, that means $\Delta p_i = 0$ holds.

2. The function $p : \Omega \to \mathbb{R}$ is piecewise defined by

$$p(r,\theta)\big|_{\Omega_i} = p_i(r,\theta), \qquad (i = 1 \dots 4).$$

Which conditions must be valid at the intersections between subdomains

$$\Omega_1 \bigcap \Omega_2, \quad \Omega_2 \bigcap \Omega_3, \quad \Omega_3 \bigcap \Omega_4, \quad \Omega_4 \bigcap \Omega_1,$$

for p to fulfil the physical requirements of the conservation law of the heat transport?

3. *(Bonus)* Determine explicit (using Matlab, Maple, Mathematica or your own program) the coefficients a_i, b_i, δ for fixed $\delta = 0.5354409455$.

(5 (+ 2) Points)

Exercise 3 *Simulation of discrete spring system*

In the lecture, the equation for the total energy stored in the system at state u was derived

$$J^{(n)}(u) = J^{(n)}_{\text{el}} + J^{(n)}_{\text{f}} = \sum_{i=0}^{n} \frac{\kappa_i}{2}(\|u_{i+1} - u_i\|) - l_i)^2 - \sum_{i=1}^{n} u_i \cdot f_i$$

where $J^{(n)} : U \to \mathbb{R}$ and

$$U = \underbrace{\mathbb{R}^3 \times \mathbb{R}^3 \times \cdots \times \mathbb{R}^3}_{n+1 \text{ times}}.$$

This corresponds to a discrete approximation of the elastic and potential energy (see lecture for details).

In this exercise, the solution $u \in \mathbb{R}^{3(n+1)}$ of the discrete energy functional will be determined numerically.

The functional fulfills the inequality

$$J^{(n)}(u) \le J^{(n)}(v) \qquad \forall v \in U.$$

To find a minimum of the functional $J^{(n)}(u)$, the nonlinear algebraic equation

$$\nabla J^{(n)}(u) = 0$$

should be solved.

It holds:

$$\frac{\partial J(u)}{\partial (u_k)_l} = \kappa_{k-1}(\|u_k - u_{k-1}\| - l_{k-1})\frac{(u_k)_l - (u_{k-1})_l}{\|u_k - u_{k-1}\|} + \kappa_k(\|u_{k+1} - u_k\| - l_k)\frac{(u_k)_l - (u_{k+1})_l}{\|u_{k+1} - u_k\|} - (f_k)_l.$$

In *dune-npde* module in directory *dune-npde/uebungen/uebung02* you can find a program, which is able to compute almost all steps which are necessary to solve the problem.

The nonlinear problem should be solved by an iterative scheme:

$$\frac{\partial J(u^i, u^{i-1})}{\partial (u_k)_l} = \kappa_{k-1}(\|u_k^{i-1} - u_{k-1}^{i-1}\| - l_{k-1})\frac{(u_k^i)_l - (u_{k-1}^i)_l}{\|u_k^{i-1} - u_{k-1}^{i-1}\|} + \kappa_k(\|u_{k+1}^{i-1} - u_k^{i-1}\| - l_k)\frac{(u_k^i)_l - (u_{k+1}^i)_l}{\|u_{k+1}^{i-1} - u_k^{i-1}\|} - (f_k)_l.$$

The iterative scheme starts with an initial value $u^0 \in \mathbb{R}^{3(n+1)}$. In each iteration a linear problem to determine u^i must be solved. Only the functions `assembleMatrix(..)` and `assembleRhs(..)`, which assemble the matrix and the right hand side of the linear problem, need to be implemented properly.

1. Complete the implementation and test it. The program is configured with the file *uebung02.ini*. The initial values correspond to a silicone-rubber fibre with a cross-section surface of 1 square millimeter. The fibre was stretched to a length of 2.5 times the initial length.
2. Test your solution and extend the program in a way that:
   - output contains y-coordinates of the spring-nodes
   - determine the mean and minimum values of y-coordinates
3. *(Bonus):* Do NOT use any *conditionals* in the matrix-iterator loop, that means the instructions which can create some jumps in compiled code (`if`, `switch`, `?:`, `std::max(..)`, etc.).

Hint: Use the function *Dune::printmatrix* for debugging. Use the DUNE documentation to find out what arguments it receives. (As the first argument, you can simply put *std::cout*)

(10 (+ 3) Points)