University of Konstanz
Department Mathematics and Statistics
L. Mechelli, J. Lu, Prof. Dr. S. Volkwein

Winter Semester 2017/2018

Exercises for
Theory and Numerics of Partial Differential Equations

`http://www.math.uni-konstanz.de/numerik/personen/mechelli/teaching.html`

Exercise Sheet 9

Submission[1]: 18^{th} January at 10:00

Exercise 1. (Theory, 10 points)
Consider the following Boundary Value Problem:

$$\begin{cases} -\Delta u = f & \text{on } \Omega, \\ u = g & \text{on } \Gamma_1, \\ \frac{\partial u}{\partial n} = 0 & \text{on } \Gamma_2 \end{cases} \tag{1}$$

where Ω is a bounded domain of $\mathbb{R}^2$ with boundary $\partial\Omega = \Gamma_1 \cup \Gamma_2$, $g \in C^2(\Gamma_1)$ and $f \in C(\overline{\Omega})$. Moreover, let

$$D_\psi = \left\{ u \in C(\overline{\Omega}) | u = \psi \text{ on } \Gamma_1 \right\}$$

and consider $\overline{u} \in C^2(\overline{\Omega}) \cap D_g$.
Prove the equivalence of these three following statements:

i) $\overline{u}$ is a stationary point of the functional $I : V_g \to \mathbb{R}$,

$$I(u) = \int_\Omega \left(\frac{1}{2} |\nabla u|^2 - fu \right) \mathrm{d}x\mathrm{d}y$$

where $V_g = H^1(\Omega) \cap D_g$

ii) $u = \overline{u} \in V_g$ satisfies

$$\int_\Omega (\nabla u \cdot \nabla v - fv)\, \mathrm{d}x\mathrm{d}y = 0$$

for all $v \in V_0$

iii) $\overline{u}$ solves the Boundary Value Problem (1)

Hints:

1. For proving the equivalence "i)$\Leftrightarrow$ii)" compute

$$\frac{\partial}{\partial \varepsilon} I(\overline{u} + \varepsilon v) \Big|_{\varepsilon=0}$$

2. For $v \in H^2(\Omega)$ and $w \in H^1(\Omega)$ holds:

$$\int_\Omega \nabla v \cdot \nabla w \mathrm{d}x\mathrm{d}y = -\int_\Omega \Delta v w \mathrm{d}x\mathrm{d}y + \int_{\partial\Omega} \frac{\partial v}{\partial n} w dS,$$

where n is the outward-pointing unit normal of Ω. This generalized Green formula for H^1 function has not to be proved and can be used in the exercise.

Exercise 2. (Programming, 6 points)
In this exercise, you are supposed to download on the above url the correct `compute_fd_grid` of Exercise 8.4 and modify it, in order to include some indexing which will become important in later programs. This means that your function should after this exercise return some lists *in addition* to the ones it has computed in Exercise 8.4. In order to illustrate this exercise in detail, consider the following simple grid as an example:

[1] The Theory Exercises will be collected at the begin of the lecture. The Programming Exercises have to be sent by email to `luca.mechelli@uni-konstanz.de` (group of Tuesday) and to `hai.nguyen-pham@uni-konstanz.de` (group of Wednesday) **before** the submission's deadline.

13	14	15		12		13
10	11	12		9	10	11
7	8	9		6	7	8
4	5	6		3	4	5
1	2	3		1		2

Table 1: Example grid with rectangle point indices (left) and domain point indices (right). This example is constructed in the `main_9_2.m` file by the name `grid0`

i. After the computation of `grid.X1`, `grid.X2`, `grid.Nodes_List` and `grid.Num_Nodes` for the rectangle $[a_1, b_1] \times [a_2, b_2]$ in Exercise 8.4 and, thus, before the elimination phase (part iv. of Exercise 8.4), create two lists `grid.listX2P` and `grid.listP2X` mapping the indices of the rectangle grid points between the `X1`&`X2`-indexing and the `grid.Num_Nodes`-indexing. For the example grid above, this would look like this:

$$\texttt{listX2P} = \begin{bmatrix} 1 & 4 & 7 & 10 & 13 \\ 2 & 5 & 8 & 11 & 14 \\ 3 & 6 & 9 & 12 & 15 \end{bmatrix} \qquad \texttt{listP2X} = \begin{bmatrix} 1 & 2 & 3 & 1 & 2 & 3 & 1 & 2 & 3 & 1 & 2 & 3 & 1 & 2 & 3 \\ 1 & 1 & 1 & 2 & 2 & 2 & 3 & 3 & 3 & 4 & 4 & 4 & 5 & 5 & 5 \end{bmatrix}$$

(**Hint**: the command `repmat` and `reshape` could be useful).

ii. Using the above list, create a list `grid.neighbours` such that `grid.neighbours(:,i)` contains the rectangle indices of the neighbours of the i-th rectangle point in the order west, east, north and south. If the neighbour does not exist, fill in 0. In the above example, this would yield

$$\texttt{neighbours} = \begin{bmatrix} 0 & 1 & 2 & 0 & 4 & 5 & 0 & 7 & 8 & 0 & 10 & 11 & 0 & 13 & 14 \\ 2 & 3 & 0 & 5 & 6 & 0 & 8 & 9 & 0 & 11 & 12 & 0 & 14 & 15 & 0 \\ 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \end{bmatrix}$$

iii. During the elimination phase (part iv.) of Exercise 8.4, remove the grid points also in `grid.neighbours`, `grid.listX2P` and `grid.listP2X`. Then, transform the indexing of the nodes according to the new order. (Compare black rectangle point indexing and blue domain indexing in Table 1). Adapt to this new indexes `grid.neighbours`, `grid.listX2P` and `grid.listP2X`. In the above example, this should result in

$$\texttt{neighbours} = \begin{bmatrix} 0 & 0 & 0 & 3 & 4 & 0 & 6 & 7 & 0 & 9 & 10 & 0 & 0 \\ 0 & 0 & 4 & 5 & 0 & 7 & 8 & 0 & 10 & 11 & 0 & 0 & 0 \\ 3 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 0 & 13 & 0 & 0 \\ 0 & 0 & 1 & 0 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 11 \end{bmatrix}$$

$$\texttt{listX2P} = \begin{bmatrix} 1 & 3 & 6 & 9 & 12 \\ 0 & 4 & 7 & 10 & 0 \\ 2 & 5 & 8 & 11 & 13 \end{bmatrix} \qquad \texttt{listP2X} = \begin{bmatrix} 1 & 3 & 1 & 2 & 3 & 1 & 2 & 3 & 1 & 2 & 3 & 1 & 3 \\ 1 & 1 & 2 & 2 & 2 & 3 & 3 & 3 & 4 & 4 & 4 & 5 & 5 \end{bmatrix}$$

iv. Identify the boundary points in the grid. A boundary point is defined as a point that does not have all four neighbours. Store the information again in two lists `grid.listB2P`, containing the indexes of the boundary points, and `grid.listP2B`, that has a 1 if the point in the i-th position is a boundary point or a 0 otherwise. These two lists would take the following form in the example:

$$\texttt{listB2P} = \begin{bmatrix} 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 8 \ 9 \ 10 \ 11 \ 12 \ 13 \end{bmatrix}, \qquad \texttt{listP2B} = \begin{bmatrix} 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \end{bmatrix}$$

(**Hint**: the command `all` and `find` could be useful).

After this exercise, your function `compute_fd_grid`, in comparison the the one of Exercise 8.4, should also add the following fields to the output variable `grid`: `listX2P`, `listP2X`, `neighbours`, `listP2B` and `listB2P`. Test your function by calling the script `main_9_2`.

Exercise 3. (Programming, 4 points)
In this exercise, you are supposed to write a function that build the linear system generated by the 5-Point Finite Difference Scheme for the Laplace equation:

$$\begin{cases} -\Delta u = f & \text{on } \Omega \\ u = g & \text{on } \partial\Omega \end{cases}$$

with $\Omega \subset \mathbb{R}^2$. The function is called in the given `main_9_3` in the following way:

```
[A,b] = build_linear_system(grid,f,g)
```

As you can see, `build_linear_system` takes three arguments:

1. `grid` is the structure generated using the function `compute_fd_grid` from Exercise 9.2
2. `f` and `g` belong to the class `function_handle` and return, respectively, the value of $\texttt{f}_{ij} = \texttt{f(x1(i),x2(j))}$ and $\texttt{g}_{ij} = \texttt{g(x1(i),x2(j))}$, where `(x1(i),x2(j))` is a node of the grid.

and gives as outputs `A` and `b`, which are, respectively, the matrix and the right-hand side of the linear system. Test your function with the two examples contained in `main_9_3`:

1. **Rectangular Domain:**

$$\Omega = [0,1] \times [0,1]$$

$$f(x_1, x_2) = 8\pi^2 \sin(2\pi x_1)\cos(2\pi x_2)$$

$$g(x_1, x_2) = \sin(2\pi x_1)\cos(2\pi x_2)$$

2. **Elliptic Domain:**

$$\Omega = \left\{ (x_1, x_2) \in \mathbb{R}^2,\ \frac{x_1^2}{4} + \frac{(x_2-5)^2}{9} \leq 1 \right\}$$

$$f(x_1, x_2) = \widehat{f}(z) = \pi^2 \sin(2\pi z) - \frac{13}{9}\pi\cos(2\pi z) \quad \text{with } z = \frac{x_1^2}{4} + \frac{(x_2-5)^2}{9}$$

$$g(x_1, x_2) = 0$$