University of Konstanz
Department Mathematics and Statistics
D. Beermann, L. Mechelli, Prof. Dr. S. Volkwein

Winter semester 2016/2017
Last update: 15/12/16, 10:40

# Exercises for
# Theory and Numerics of Partial Differential Equations

`http://www.math.uni-konstanz.de/numerik/personen/beermann/en/teaching`

## Sheet 8

**Deadline: Thursday, 12/01, 5pm**

**Exercise 8.1** (Matlab) (8 points)

**Assorted files**: `main81.m` and `plot_grid.m`
Please follow the *programming guidelines* that can be downloaded under the above url.

In this exercise, you are supposed to refine the function `compute_fd_grid` from Exercise 7.1 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 7.1. In order to illustrate the procedure in detail, consider the following simple grid as an example:

| | | | | | | |
|---|---|---|---|---|---|---|
| 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 `main81.m` file by the name `grid0`

1. After the computation of `X1`, `X2` and `p` for the rectangle $[\underline{x}_1, \overline{x}_1] \times [\underline{x}_2, \overline{x}_2]$ in Exercise 7.1, create two lists `listX2P` and `listP2X` mapping the indices of the rectangle grid points between the `X1`&`X2`-indexing and the `p`-indexing. For the example grid above, this would look like this:

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

2. Using the above list, create a list `neighbours` such that `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

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

3. During the elimination phase (part 2.) of Exercise 7.1, transform the above `neighbour` list to accomodate the now-changed indexing (compare black rectangle point indexing and blue domain indexing in Table 1). In the above example, this should result in

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

Also update the lists listX2P and listP2X in the process accordingly.

4. 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 listB2P and listP2B which would take the following form in the above example:

$$\texttt{listB2P} = \big[1\ 2\ 3\ 4\ 5\ 6\ 8\ 9\ 10\ 11\ 12\ 13\big], \qquad \texttt{listP2B} = \big[1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 1\ 1\big]$$

After this exercise, your function compute_fd_grid should add the following fields to the output variable grid: listX2P, listP2X, neighbours, listP2B and listB2P. Test your function by calling the script main81.m.

**Exercise 8.2** (Theory) (6 points)

Let $[\underline{x}, \overline{x}] \subset \mathbb{R}$ be an interval which is dicretized in an equidistant manner, i.e. we have a grid

$$[\underline{x}, \overline{x}]_h = [\underline{x}, \overline{x}] \cap h\mathbb{Z} = \{x_0, x_1, ..., x_{N-1}, x_{N+1}\} \quad x_i = \underline{x} + ih \quad (i = 0, ..., N+1), \quad \overline{x} = x_{N+1}$$

where $h > 0$ is a step size and $N \in \mathbb{N}$. Let us define the Banach spaces

$$X := \Big\{ f : [\underline{x}, \overline{x}] \to \mathbb{R},\ f \text{ is continuous} \Big\} = C^0([\underline{x}, \overline{x}]) \qquad \|f\|_X := \max_{x \in [\underline{x}, \overline{x}]} |f(x)|$$

$$X_h := \Big\{ f_h : [\underline{x}, \overline{x}]_h \to \mathbb{R} \Big\} \qquad \|f_h\|_{X_h} := \max_{i=0,...,N+1} |f_h(x_i)|$$

For a function $f_h \in X_h$, we define $\|f_h\|_h := \max_{i=1,...,N} |f_h(x_i)|$

1. Prove that $\|\cdot\|_h : X_h \to \mathbb{R}$ is a seminorm on $X_h$.

2. We define the canonical embedding

$$D_h : X \to X_h, \quad (D_h f)(x_i) := f(x_i) \quad (i = 0, ..., N+1)$$

Show that $D_h \in L(X, X_h)$ with $\|D_h\|_{L(X,X_h)} = 1$

3. Computing the derivative of a function $f \in C^1([\underline{x}, \overline{x}])$ can be thought of as applying an operator

$$\Lambda :\ C^1([\underline{x}, \overline{x}]) \to C([\underline{x}, \overline{x}]), \quad (\Lambda f)(x) := f'(x) \quad \text{for all } x \in [\underline{x}, \overline{x}]$$

You have already heard of methods for numerical derivatives which we can formulate in a similar way. Take for example the *Forward Euler method*:

$$\Lambda_h^+ :\ X_h \to X_h, \quad (\Lambda_h^+ f_h)(x_i) = \frac{1}{h}\left(f_h(x_{i+1}) - f_h(x_i)\right), \quad i = 0, ..., N, \quad (\Lambda_h^+ f_h)(x_{N+1}) = 0$$

We are now interested in computing the *consistency error* of this approximation. Prove that it holds

$$\|D_h \Lambda f - \Lambda_h^+ D_h f\|_h \ \le\ C_f h \qquad \text{for all } f \in C^2([\underline{x}, \overline{x}]) \tag{1}$$

with a constant $C_f > 0$ depending on $f$.[1].

4. At last, we consider second derivatives as described by the operator

$$\Delta : C^2([\underline{x}, \overline{x}]) \to C([\underline{x}, \overline{x}]), \quad (\Delta f)(x) := f''(x) \quad \text{for all } x \in [\underline{x}, \overline{x}]$$

The standard numerical approximation of this is called the *Discrete Laplacian*

$$\Delta_h : X_h \to X_h, \quad (\Delta_h f_h)(x_i) = \begin{cases} \frac{1}{h^2}\left(f(x_{i-1}) - 2f(x_i) + f(x_{i+1})\right), & (i = 1, ..., N) \\ 0, & (i = 0, N+1) \end{cases}$$

Prove that the consistency error of this approximation is of quadratic order:

$$\|D_h \Delta f - \Delta_h D_h f\|_h \ \le\ C_f h^2 \qquad \text{for all } f \in C^4([\underline{x}, \overline{x}]) \tag{2}$$

with a constant $C_f > 0$ depending on $f$.

[1]Another way to write this would to use the Landau symbol: $\|D_h \Lambda f - \Lambda_h^+ D_h f\|_h = \mathcal{O}(h)$, implying linear convergence as $h \to 0$. One can also show that the implicit Euler method is likewise linear and the central difference scheme $(\Lambda_h^c f_h)(x_i) = \frac{1}{2h}\left(f_h(x_{i+1}) - f_h(x_{i-1})\right)$ is of quadratic convergence for three times differentiable functions.

**Exercise 8.3** (Theory) (6 points)

We consider a one-dimensional stationary *advection-diffusion-reaction* equation:

$$-\underbrace{u''(x)}_{\text{Diffusion}}+\underbrace{b(x)u'(x)}_{\text{Advection}}+\underbrace{c(x)u(x)}_{\text{Reaction}}=\underbrace{f(x)}_{\text{Source term}} \quad \text{for all } x\in(0,1)$$
$$u(0)=u(1)=0 \tag{3}$$

$u$ describes the steady state of a substance contained in a flowing medium. The diffusion term describes the local spreading of the substance, the advection term represents the transport from the medium flow, the reaction term accounts for the reaction between substance and medium, leading to an increase or a decrease of the substance and the source term allows for external injection or extraction of the substance.

1. Discretize (3) by introducing an equidistant grid as described in Exercise 8.2. Use the *Discrete Laplacian* for the diffusion term and the *central difference scheme* for the advection term. Write the grid function $u_h \in X_h$ as

$$\begin{pmatrix} u_h(x_0)\\ u_h(x_1)\\ \vdots \\ u_h(x_N)\\ u_h(x_{N+1})\end{pmatrix}=\begin{pmatrix}0\\ y_1\\ \vdots\\ y_N\\ 0\end{pmatrix}$$

and write the discretized version of (3) as a linear problem $Ay=r$.

2. A matrix $B=(B_{ij})_{i,j=1,\dots,N}\in\mathbb{R}^{N\times N}$ is called *strictly diagonally dominant* if $|B_{ii}|>\sum_{\substack{i=1\\ i\neq j}}^{N}|B_{ij}|$. It can be shown that a strictly diagonally dominant matrix is always regular. When is the matrix $A$ from above strictly diagonally dominant and thus the discretization performed in 1. well-defined?

3. In 2., you should have realized that the advection coefficient $b$ in (3) may cause problems for the discretization. Now, repeat 1. by using - instead of the central difference scheme - a so-called *streamline-upwind difference scheme* for the discretization of the advection part:

$$\Lambda_h^{SU}:X_h\to X_h,\quad (\Lambda_h^{SU}u_h)(x_i)=\begin{cases}\frac{1}{h}\left(u(x_{i+1})-u(x_i)\right), & i=1,...,N,\ \text{if } b(x_i)<0\\ \frac{1}{h}\left(u(x_i)-u(x_{i-1})\right), & i=1,...,N,\ \text{if } b(x_i)\geq 0\\ 0, & i=0,N+1\end{cases}$$

4. If we denote the problem resulting from 3. by $A^{SU}y=r^{SU}$, check again for diagonal dominance as in 2. What do you observe?