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 11

Submission[1]: 1^{st} February at 10:00

Exercise 1. (Theory, 10 points)
Let $\Omega = (0,1) \subset \mathbb{R}$ be discretized in the usual equidistant manner with step size $h > 0$:

$$0 = x_1 < \ldots < x_N = 1, \quad x_i = ih \ (i = 1, ..., N)$$

We define the space of piecewise linear Finite Elements:

$$X_h := \left\{ f : [0,1] \to \mathbb{R}, \middle|\, f \text{ continuous, } f|_{[x_i, x_{i+1}]} \text{ affine linear for } i = 1, ..., N-1 \right\}$$

1. Show that the *nodal elements* given by $\phi_i \in X_h$, $\phi_i(x_j) = \delta_{ij}$ $(i, j = 1, ..., N)$ are well-defined and form a basis of X_h.
2. Show that $X_h \subset H^1(\Omega)$ by computing and proving the concrete gradients $\nabla \phi_i$ $(i = 1, ..., N)$.
3. Derive the Galerkin discretization of the boundary value problem

$$\begin{aligned} -u''(x) &= f(x) \quad \text{for } x \in \Omega \\ \frac{\partial u}{\partial n}(x) + u(x) &= 0 \qquad \text{for } x \in \partial\Omega \end{aligned} \tag{1}$$

   using the Finite Element space X_h as both test and ansatz space. Here, $f : (0,1) \to \mathbb{R}$ is a given inhomogeneity. You should end up with an algebraic system of the type $Ay = b$.

Exercise 2. (Programming, 10 points)
Consider the following problem:

$$\begin{cases} -\lambda \Delta u + cu = f & \text{in } \Omega \\ \frac{\partial u}{\partial n} = 0 & \text{on } \partial\Omega \end{cases} \tag{2}$$

with

$$\Omega = \left\{ (x, y) \in \mathbb{R}^2 : \frac{x^2}{a^2} + \frac{y^2}{b^2} \leq 1 \right\},$$

$f \in L^2(\Omega)$ and $\lambda, c, a, b > 0$. For implementation purposes, declare a and b as `global` parameters in your main script.[2] We will utilize commands from the PDE Toolbox to generate the triangulation $\mathcal{T}$ of Ω. Proceed as follows in order to write the main file `main_11_2`:

0. **PDE model creation:** Create the structure `model` with the MATLAB command `createpde`.
1. **Geometry implementation:** Write a function `geometryFunction.m` to describe the geometry of Ω by using a suitable analytical boundary representation. For information about how such a function should look like, see

   http://de.mathworks.com/help/pde/ug/create-geometry-using-a-geometry-function.html

   Especially focus on the circle's example. Then use the command `geometryFromEdges` to create the `geometry` structure and `pdegplot` to test your results.

[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.

[2] For information on global variables, see `https://de.mathworks.com/help/matlab/ref/global.html`

The following point 2. should be solved in the script `main_11_2`.
Do not use point-and-click for these!

2. **Mesh generation:** Generate a mesh with a maximum element size of 0.1 and visualize the mesh. The following webpages could be helpful
`http://de.mathworks.com/help/pde/ug/pde.pdemodel.generatemesh.html?s_tid=doc_ta`
and
`http://de.mathworks.com/help/pde/ug/pdemesh.html?searchHighlight=pdemesh&s_tid=doc_srchtitle`

Download from the above url the guidelines for the next part.

3. **Build finite element matrices:** This is supposed to be done in a function `ComputeFiniteElement(mesh,f)`, which takes in input the mesh generated at step 2. and the PDE right-hand side `f` in a function handle format and gives in output the mass matrix M, the stiffness matrix A and the finite element vector F for the right-hand side.

4. **Solve the linear system:** Convert the finite elements matrices in a sparse form and solve the discrete linear system with the backslash command.

5. **Plot the solution:** Plot the solution in a 3D plot, using the MATLAB command `pdeplot`.

Text your code with the right-hand side functions f of exercise 10.2 of the previous exercise sheet.