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

Winter semester 2016/2017

Exercises for
Theory and Numerics of Partial Differential Equations

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

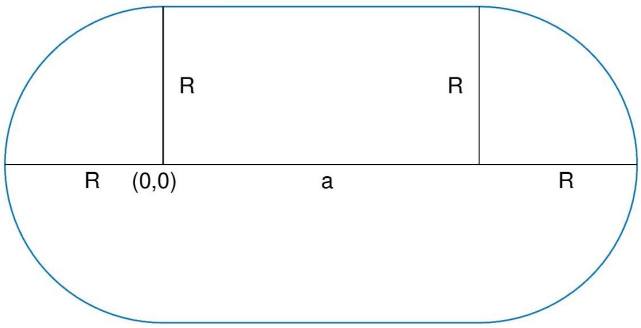
Sheet 10

Deadline: Thursday, 26/01, 3:30pm

Exercise 10.1 (Matlab) (10 points)

Please follow the *programming guidelines* that can be download under the above url.

This exercise is intended for you to familiarize yourself with a useful Matlab application called the *PDE Toolbox*. It is a Finite Element (FE) package to solve some elliptic and parabolic PDEs. As an example, consider the following parameter-dependent domain:



and the elliptic PDE:

$$-\Delta y(x) + \lambda y(x) = b \quad \text{for all } x \in \Omega, \qquad \frac{\partial y}{\partial n}(x) + \alpha y(x) = c \quad \text{for all } x \in \partial\Omega \tag{1}$$

where $\Omega \subset \mathbb{R}^2$ is given by the interior of the blue line. It depends on the parameters $R > 0$, $a > 0$, $\lambda, b, \alpha, c \in \mathbb{R}$. For implementation purposes, declare R and a as `global` parameters in your main script and make them available in each function.[1] We will utilize commands from the PDE Toolbox to treat (1) numerically.

0. **Familiarizing**: For an excellent overview of the workflow with the PDE toolbox, see

   http://de.mathworks.com/help/pde/ug/solve-problems-using-pdemodel-objects.html

   Familiarize yourself with the idea of how to solve these kinds of problems.

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 various way that this function will be called by the PDE toolbox (0,1,2 inputs, `bs` scalar or a vector, ...). Then use the command `pdegplot('geometryFunction')` to test your results.

The following two points should be solved in a script.
Do not use point-and-click for these!

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

2. **PDE specification:** Specify the PDE coefficients in (1) and generate a mesh with a maximum element size of 0.05. Visualize the mesh.

3. **PDE solving:** Solve the problem for various combinations of λ, b and c. Visualize the solution y and its gradient using the `subplot` command (y on the top, ∇y on the bottom). Do all this by writing and calling an external function `solve_elliptic_problem`. Write a thorough report documenting how the variation of λ, b, α and c affects the solution y.[2]

Exercise 10.2 (Theory) (3+3+4=10 points)

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

$$0 = x_0 < \ldots < x_{N+1} = 1, \quad x_i = ih \ (i = 0, ..., N+1)$$

We define the space of piecewise linear Finite Elements:

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

1. Show that the *nodal elements* given by $\phi_i \in X_h$, $\phi_i(x_j) = \delta_{ij}$ $(i, j = 0, ..., N+1)$ 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 $\nabla \phi_i$ $(i = 0, ..., N+1)$.

3. Derive the Galerkin discretization of the boundary value problem

$$\begin{aligned} -\Delta y(x) &= f(x) && \text{for } x \in \Omega \\ \frac{\partial y}{\partial n}(x) + y(x) &= 0 && \text{for } x \in \partial\Omega \end{aligned} \tag{2}$$

   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 $Ay = b$.

[2] If you see very strange solutions for some combinations, it might not be the fault of your code ;-)