Fachbereich Mathematik & Informatik
Freie Universität Berlin
Prof. Dr. Carsten Gräser, Tobias Kies

7th exercise for the lecture

Numerics IV

Winter Term 2016/2017

http://numerik.mi.fu-berlin.de/wiki/WS_2016/NumericsIV.php

Due: Tuesday, Dec 13th, 2016 for theoretical exercises and Tuesday, Jan 3rd, 2017 for programming exercises

Exercise 1 (4 TP)
Let $\Omega \subseteq \mathbb{R}^n$, $f \in L^2(\Omega)$ and

$$J(u) := \int_\Omega \sqrt{1 + \|\nabla u(x)\|^2}\,\mathrm{d}x - \int_\Omega f(x)u(x)\,\mathrm{d}x$$

for all $u \in H^1(\Omega)$. Let $u, v, w \in H^1(\Omega)$. Prove that

$$\begin{aligned} DJ(u;v) &= \int_\Omega \frac{\nabla u(x)\cdot\nabla v(x)}{\sqrt{1+\|\nabla u(x)\|^2}} - f(x)v(x)\,\mathrm{d}x \\ D^2J(u;v,w) &= \int_\Omega \frac{\nabla v(x)\cdot\nabla w(x)}{\sqrt{1+\|\nabla u(x)\|^2}} - \frac{(\nabla u(x)\cdot\nabla v(x))(\nabla u(x)\cdot\nabla w(x))}{(1+\|\nabla u(x)\|^2)^{3/2}}\,\mathrm{d}x. \end{aligned}$$

Here $DJ(u;v)$ denotes the Gâteaux-derivative of J in u in direction v and $D^2J(u;v,w)$ is the second Gâteaux-derivative of J in u in the directions v and w, i. e.

$$D^2J(u;v,w) = \lim_{0\neq t\to 0} \frac{1}{t}\left(DJ(u+tw;v) - DJ(u;v)\right).$$

Reminder: Don't forget the exercise from the last exercise sheet:

Exercise 2 (4 TP)
Let $H = H_0^1(\Omega)$ with $\Omega \subseteq \mathbb{R}^n$ bounded with smooth boundary and define

$$J\colon H_0^1(\Omega) \to \mathbb{R}, \qquad v \mapsto \frac{1}{2}\int_\Omega \varepsilon\|\nabla u(x)\|^2 + \frac{1}{\varepsilon}\Psi(u(x))\,\mathrm{d}x$$

where $\varepsilon \in \mathbb{R}_{>0}$ and $\Psi\colon \mathbb{R} \to \mathbb{R}$ such that $\Psi \circ v \in L^1(\Omega)$ for all $v \in H_0^1(\Omega)$. Suppose $u \in H_0^1(\Omega) \cap H^2(\Omega)$ and assume that $\Psi \in C^1(\mathbb{R})$ with $\Psi' \circ u \in L^2(\Omega)$. Compute the $L^2(\Omega)$-gradient of J in u. How does this relate to the Allen-Cahn equation?

Please turn over...

Programming exercises.
The following exercises will allow you to collect programming points (PP) instead of the usual theory points (TP).

Exercise 3 (0 PP)
Download the Matlab PDE-utils package from Git. You can do so by either directly downloading it from

https://git.imp.fu-berlin.de/graeser/matlab-pdeutils/repository/archive.zip

or by cloning the git repository onto your hard drive by issuing the command

```
git clone git@git.imp.fu-berlin.de:graeser/matlab-pdeutils.git
```

in your Linux-terminal. Note that `git` is usually installed by default on Linux devices. There also exists a version for Microsoft Windows (see https://git-scm.com/download/win).

Exercise 4 (0 PP)
This exercise helps you to get you acquainted with `pdeutils`. Let us consider the Poisson equation with Dirichlet boundary conditions

$$-\Delta u = f \text{ on } \Omega, \qquad u = g \text{ on } \partial\Omega$$

for some polyhedral domain $\Omega \subseteq \mathbb{R}^n$. From the lectures we know that the solution to this problem is approximated by finding a $u_h \in S_h$ with $u_h|_{\partial\Omega} = I_h g$ and

$$\forall v_h \in S_h \cap H_0^1(\Omega)\colon \int_\Omega \nabla u_h \cdot \nabla v_h \,\mathrm{d}x = \int_\Omega f v \,\mathrm{d}x.$$

Here $S_h \subseteq H^1(\Omega)$ is a finite element space with respect to a grid $\mathcal{T} = \bigcup_{T\in\mathcal{T}} T$. By the function $I_h\colon H^1(\Omega) \to S_h$ we denote a suitable approximation operator (in the following we assume that I_h simply is the interpolation operator).
Now, suppose that we are given a basis $\{\psi_i\}_{i=1,\dots,N}$ of S_h. Then it is possible to write $u_h = \sum_{i=1}^N u_i\psi_i$ where the u_i are the coefficients of $u_h \in S_h$ with respect to the basis $\{\psi_i\}_{i=1,\dots,N}$. The vector of these coefficients shall be denoted by $\overline{u} := (u_i)_{i=1,\dots,N} \in \mathbb{R}^n$. Let us denote by $O \subseteq \{1,\dots,N\}$ the indices for which the associated nodes x_i of the grid $\mathcal{T}$ lie on the boundary $\partial\Omega$. Conversely, by $I = \{1,\dots,N\}$ denote the nodes of the grid $\mathcal{T}$ that lie in the interior of Ω. By inserting the representation $u_h = \sum_{i=1}^N u_i\psi_i$ into the above variational equation and by using $u_i = f(x_i)$ for $i \in O$ we obtain the equations

$$\forall i \in I\colon \qquad \sum_{j=1}^N u_j \int_\Omega \nabla\psi_j \cdot \nabla\psi_i \,\mathrm{d}x = \int_\Omega f\psi_i \,\mathrm{d}x$$

$$\forall i \in O\colon \qquad \sum_{j=1}^N u_j\delta_{ij} = f(x_i)$$

This system can be written as a linear equation

$$A\overline{u} = b$$

where $A_{ij} = \int_\Omega \nabla\psi_i \cdot \nabla\psi_j \,\mathrm{d}x$ for $i \in I$ and $j \in I \cup O$ and $A_{ij} = \delta_{ij}$ for $i \in O$ and $j \in I \cup O$, and $b_i = \int_\Omega f\psi_i \,\mathrm{d}x$ for $i \in I$ and $b_i = f(x_i)$ for $i \in O$.
Our goal is to solve the above linear system. We encountered the following essential ingredients:

- The grid $\mathcal{T}$ that discretizes our domain Ω.
- The finite element space S_h. The space S_h usually consists out of piecewise polynomials with respect to the grid $\mathcal{T}$, and so S_h depends on $\mathcal{T}$.
- The matrix A and the vector b. Those require integration of some integrals which depend on the finite element space S_h and the basis that we chose therein (and of the right hand side f). Since the $T \in \mathcal{T}$ have simple geometries and the S_h are piecewise polynomial it is possible to evaluate these integrals exactly by using relatively simple quadrature rules for each element $T \in \mathcal{T}$.
- Finally, we need to solve the system $A\overline{u} = b$. This can be done by using `Matlab`'s backslash operator \, for example.

A possible algorithm for solving the Poisson equation can be stated as:

1. Given data: f, g and Ω.
2. Create a grid $\mathcal{T}$ of Ω. Let's assume that all $T \in \mathcal{T}$ are triangles.
3. Specify a finite element space S_h over $\mathcal{T}$. This is done by specifying a *local reference basis* with respect to a reference triangle $\hat{T}$. In case of piecewise linear finite elements in 2D we could for example have $\hat{T} = \{(x, y) \in [0, 1]^2 \mid y \leq 1 - x\}$ and
$$\hat{\psi}_1(x, y) = 1 - x - y, \qquad \hat{\psi}_2(x, y) = x, \qquad \hat{\psi}_3(x, y) = y.$$
4. Iterate over all elements $T \in \mathcal{T}$ and compute the integrals for A_{ij} and b_i. This can be done by transforming T onto the reference element $\hat{T}$ and then evaluating the integrals with respect to the local basis on the reference element.
5. Modify A and b such that they incorporate Dirichlet boundary values.
6. Solve $Au = b$, for example by `u = A\b`.

Having this in mind, look at the example function `pdeutils_examples/fe_poisson.m`. Read the code (and the functions that it invokes) and try to understand

- how a basis is created,
- how the system is assembled, especially how global and local assemblers work,
- how the boundary conditions are incorporated,
- and how the system is solved.

Also look at the functions in `pdeutils/grid` for grid generation.

Please turn over...

Exercise 5 (8 PP)
Find and compute an example for the Poisson equation and compare it with its exact solution, i. e. do the following: Choose a bounded domain $\Omega \subseteq \mathbb{R}^2$, $f \in L^2(\Omega) \setminus \{0\}$ and $g \in L^2(\partial\Omega) \setminus \{0\}$ for which you know an analytic expression of the exact solution u of

$$-\Delta u = f \text{ on } \Omega, \qquad u = g \text{ on } \partial\Omega.$$

Compute the discrete approximation u_h of this Poisson problem using piecewise linear finite elements (over a reasonable fine grid) and the `pdeutils` package. Compare your results with the exact solution by plotting both u_h and u.

Important! Assuming that you extracted `pdeutils` into a folder names `matlab-pdeutils`. Before you can run any commands from `pdeutils`, you have to navigate to the folder `matlab-pdeutils` and run the command

```
pdeutils_addpaths
```

in order to tell `Matlab` where the routines of `pdeutils` are located. After this you can invoke the `pdeutils` functions from anywhere.

Have fun!