Dr Andreas Dedner
Dr Björn Stinner

MA915 C2 Advanced Topics in Numerical Analysis
Informations on the Practicals

Introduction

The module *C2 MA915 Advanced Topics in Numerical Analysis - Finite Element Methods* consists to two thirds of lectures and to one third of computer sessions. Finite element methods are constructive methods to approximate solutions to pdes. The practicals serve to

- get a feeling for the problems emerging when implementing such methods,
- reproduce theoretical results presented in the lectures by means of examples.

The exercises (as well as the assignments) will be based on DUNE
`http://www.dune-project.org/dune.html`),
a powerful and general environment for mesh based numerical methods. During the practical sessions we will work on small exercises for which the solution is provided the following week. The code is also the basis for the assignments.

Installing the software

Via a simplified version you are going to be introduced to DUNE.

1. Boot Linux on your laptop.
2. From the subpage *material* of the module webpage, download the file `C2.tar.gz` by clicking on the link code behind Practical 1: *its rather large so you should have a good internet connection.*
3. Move this file into a convenient subfolder (or simply into your home folder) on the MASDOC laptop and extract the content with the command
   `tar -xzvf C2.tar.gz`
4. Call
   ```
   cd C2
   sh ./build.sh
   cd C2-practical
   ```

There is a subfolder `src` with folders `1,2,...,10`. Some C++ example files are already contained in folder `src/1` - if you have any questions concerning those then ask us. We will start with session 2 (code will be put into `src/2`) this week. Files for each session will be provided in the form of patches that you will be able to download from the module website. Patches will contain the solution for lasts week practical and some new files to work on. After downloading a patch (lets say `patch2.src`) into some directory (lets say into the `C2` folder), you can apply the patch by calling `sh ./updatePractical.sh .. 2` from the `C2-practical` folder. We will explain all this in more detail...
Instructions and further informations on the software can be obtained from the documentation that can be retrieved by calling

```
konqueror html/index.html
```

(any other browser does the job as well) in the directory `C2-practical`.

First practical: Linear interpolation in 1D (some C++ exercises as warm up)

The goal is to linearly interpolate a given function $u : [a, b] \to \mathbb{R}$ on a closed interval $[a, b] \subset \mathbb{R}$. For a number $N \in \mathbb{N}$, let $h := (b - a)/(N - 1)$ and $x_i := a + ih$, $i = 0, \ldots, N - 1$. We define a piecewise linear function $u_h : [a, b] \to \mathbb{R}$ by evaluating u in the equidistributed points x_i, setting $u_h(x_i) := u(x_i)$ and demanding u_h to be linear on each interval $[x_{i-1}, x_i]$. In order to estimate the error of the interpolation let $y_i := a + (i + 0.5)h$, $i = 0, \ldots, N - 1$, and compute

$$e_h(u) := \sum_{i=0}^{N-1} h|u(y_i) - u_h(y_i)| \quad \Big(\approx \int_a^b |u - u_h| dx \Big)$$

The files `exercise1a.c` and `exercise1a.cc` contain the a `C` and a `C++` version, respectively. Ensure that you understand the codes and that you can compile both files on your laptop. Test with the functions $u^{(1)}(x) := \sin(2\pi x)$, $u^{(2)}(x) := \sqrt{|x|}$, and $u^{(3)}(x) := x \sin(1/x)$ (extended by $u^{(3)}(0) := 0$).
The output can be visualized with the program `gnuplot` - try out!
In `exercise1b`, a `Grid` class is introduced in order to separate the geometric informations from the data. Furthermore, we see a template: The class `std::vector<double>` is kind of 'parameterized' by another class (here `double`). It is introduced to facilitate the data handling.
The problem handling and usability is improved in `exercise1c`: A virtual base class for the actual problem is used, and specific problems can be addressed from the command line whence there is no need for recompiling any more.
Towards DUNE, `exercise1d` adds an `Element` class as a more natural view of the intervals $[x_{i-1}, x_i]$ in the finite element context. It also introduces iterators for the loops over the grid. Observe that it can be compiled and linked with `make exercise1d` - it has been added to the file `Makefile.am`. Note that the default compiler flags in the Makefile lead to heavily optimized code. If you want to have information for debugging (e.g. for using `gdb` or `ddd`) use for example `make CXXFLAGS="-g -Wall" exercise1d`.

Exercise

Determine the order of convergence of $e_h(u)$ in terms of h for each of the above problems. For this purpose, compute the *experimental order of convergence*

$$eoc(h_1, h_2, u) := \frac{\log(e_{h_1}(u)) - \log(e_{h_2}(u))}{\log h_1/h_2}.$$

Extend the program so that it provides e_h, $e_{\frac{1}{2}h}$ and $eoc(h, \frac{1}{2}h, u)$ for a given start discretization h. If you want, extend the program further by computing for given h and L the sequence e_{h_l} with $h_l = 2^{-l}h$ for $l = 0, \ldots, L$ and $eoc(h_{l-1}, h_l, u)$ for $l = 1, \ldots, L$.