Introduction to the structure of the course

The course is an overview of numerical methods that are used to solve partial differential equations. The main emphasis is on covering a wide range of practical methods instead of deep numerical analysis. Minimal mathematical background is provided. If possible, the main theoretical results are illustrated by geometrical finite dimensional analogs. The implementation and some theoretical properties of the main methods are investigated by programming exercises.

The main part of learning will be hopefully obtained via doing exercises. This should also lower the barrier to use mathematical software for personal research.

About lecture notes

Beware of the text you read. It is freshly written and contains mistakes for sure. Moreover, huge mathematically relevant parts and crucial definitions are omitted! In particular, definitions of function spaces associated with treated PDEs is done in a very heuristic manner and the questions of existence and uniqueness are not touched at all. This (and other crimes) is done in order to make presentation more fluent and to concentrate strictly on the main ideas behind the methods. Additionally, math department offers excellent PDE-courses, which I recommend for everyone. However, I try to give references to good sources, where briefly mentioned theories are fully explained.

Some words about notation. Matrices and vectors are (at least they should be) written bold and matrices by capital letters. Thus, $S$ is a matrix, which has elements $S_{ij}$. However, $B_i$ is also a matrix, albeit it belongs to a set indexed by $i$. Similarly $x$ is vector, and $x_k$ is some particular vector. Components of the coordinates are denoted by $x_1$, etc, but sometimes it is very convenient to use coordinates $(x,y)$. On the other hand, $x$ may refer to a function. For example it might depend on time $x(t)$, but the argument is omitted. Experienced reader should note that unfortunately there are no explanatory pictures in the notes, which in some sections makes them painful
to read. Hopefully, beautiful blackboard drawings during the lectures and the fruitful discussions will clarify everything.

**Integration quadratures and order of convergence**

Numerical integration schemes are easy to illustrate graphically, e.g., “pillar scheme”, trapezoidal rule and midpoint rule, and Simpson rule. They all aim to approximate the definite integral using some quadrature. Quadrature consists of points \( x_k \in [a, b] \) and the corresponding weights \( w_k \). Thus,

\[
q := \int_a^b f(x) \, dx \approx \sum_{k=1}^{N} w_k x_k =: q_N
\]

The main characteristic of different quadratures is their order. The order of convergence for a method approximating the quantity \( q \) using \( N \) grid points is \( p \) in the following definition

\[
|q - q_N| < C N^{-p} \quad \text{or} \quad |q - q_N| = O(N^{-p}).
\]

Similarly for a mesh size parameter \( h = \frac{b-a}{N} \)

\[
|q - q_h| < C h^p \quad \text{or} \quad |q - q_h| = O(h^p).
\]

Similar estimates are derived in analysis and proving this type of (a priori) estimates for various numerical methods is one of the classical branches of numerical analysis.

When studying the order of some implemented method (numerical integrator, ODE-solver, PDE-solver, etc.), *loglog-graphs* are very practical and typically used. This is due to the fact that curves of the form \( y = ax^p \) are transformed into straight lines \( \log y = p \log x + \log a \) and the slope of the line shows the order.