Numerical Methods for Partial Differential Equations

Thomas Wick

Gottfried Wilhelm Leibniz Universität Hannover (LUH)

Institut für Angewandte Mathematik (IfAM)

AG Wissenschaftliches Rechnen (GWR)

Welfengarten 1, 30167 Hannover, Germany

https://www.ifam.uni-hannover.de/wick.html

https://thomaswick.org

thomas.wick@ifam.uni-hannover.de

Version 2, last update:
Friday, January 21, 2022
DOI https://doi.org/10.15488/11709

Version 1, Jan 28, 2020
DOI https://doi.org/10.15488/9248

by Elisa Wick

Foreword

These notes accompany lectures to the classes

Numerik partieller Differentialgleichungen (NumPDGL)
Numerics of partial differential equations (NumPDE)
An international open class (IntNumPDE)

at the Leibniz Universität Hannover (LUH). These classes are classical $4 + 2$ lectures (per week: two times 90 minutes lecture and one 90 minute exercise class) in the German university system. Usually, no programming or software developements are made in these classes. Rather, they are accompanied by an extra class of 2 hours

Finite-Elemente-Programmierpraktikum in C++ (FEMProg)

in which finite elements for solving partial differential equations are implemented by the students (Class 3b; see next pages).

A brief summary of all **topics** in these notes is:

- Recapitulation of characteristic **concepts of numerical mathematics**
- Brief review of **mathematical modeling** of differential equations
- **Discretization** with Finite differences (FD) for elliptic boundary value problems
- **Discretization** with Finite elements (FEM) for elliptic boundary value problems
- A posteriori **error estimation** of goal functionals using duality arguments
  (extra class; see Class 3c on the next pages)
- **Numerical solution** of the discretized problems
- A brief introduction to **vector-valued problems** (elasticity/Stokes) (if time permits)
- **Time-dependent problems:** Methods for parabolic and hyperbolic problems
- Numerical methods for **nonlinear and coupled problems** (Class 5a)

The prerequisites are lectures in calculus, linear algebra and an introduction to numerics or as well classes on the theory of ordinary (ODE) or partial (PDE) differential equations. Also, functional analysis (FA) is recommended, but not necessarily mandatory. Classes in continuum mechanics may also be helpful, but are not mandatory to understand these notes.

On the other hand, for the convenience of all of us, many results from linear algebra, analysis and functional analysis are included (often without the proofs though) in order to achieve a more or less self-contained booklet.

To fully enjoy these notes, the reader should have a taste for practical aspects and algorithms as well as he/she should be ready to dive into theoretical aspects of variational formulations and numerical analysis. In most sections, **numerical tests complement algorithms and theory**. Snippets of **programming codes** are provided as well at the end.

At this place I want to thank my students from the WS 2017/2018, WS 2019/2020 classes, the Peruvian participants from the international open class WS 2020/2021, and again in presence WS 2021/2022 and my doctoral researchers and Dr. Maryam Parvizi for letting me know about mistakes and typos. Furthermore thanks to Katrin Mang for some tikz pictures and latex improvements.

Enjoy reading and please let me know about mistakes via the email address indicated on the front page.

Hannover, January 2022 **Thomas Wick**

Specific structure and topics in a $4+2$ lecture class

As previously mentioned, such a class is typical in the German system and has usually the following characteristics:

- 10 ECTS (European Credit Transfer and Accumulation System)
- 4 hours lecture per week, i.e., two times 90 minutes lecture per week
- 2 hours exercises per week, i.e., once 90 minutes per week
- In total per semester around $26-28$ lecture meetings (each - as said - 90 minutes)
- In total per semester around $13-14$ exercise meetings (each - as said - 90 minutes)

As general info: the German winter semester usually runs from mid-October until end of January/begin February. The German summer semester usually runs from begin/mid-April until mid July. Both have usually around $12-14$ weeks lecture courses.

Topics

The topics may be[1] (L = lecture):

1. L1: Recapitulation of characteristic concepts of numerical mathematics
   Parts of Chapter 2
2. L2-L3: Brief review of mathematical modeling of differential equations
   Parts of Chapter 4
3. L4: Classifications
   Parts of Chapter 5
4. L5-L8: Finite differences (FD) for elliptic boundary value problems
   Parts of Chapter 6
5. L9-L20: Finite elements (FEM) for elliptic boundary value problems
   Major parts of Chapter 8 and partially Chapter 7
6. L21-L23: Numerical solution of the discretized problems
   Parts of Chapter 10 with a focus on multigrid
7. L24-L26/27: Methods for parabolic and hyperbolic problems (time-dependent PDEs)
   Chapter 12
8. L28: Recapitulation of contents
   Chapter 18.1

[1] Remark: the entire lecture notes contain too much additional materials. The purpose of these lecture notes was not a specific tailored booklet to some class, but rather a self-contained work that in particular also covers advanced topics such as error estimation, nonlinear, and coupled problems.

Structure of a possible numerical analysis/scientific computing cycle

1. Class 1: Introduction to numerical analysis (Numerik 1)
2. Class 1b: Algorithmisches Programmieren in `C++`
3. Class 2: Numerical methods for ODEs and eigenvalue problems (Numerik 2)
4. **Class 3: Numerical methods for PDEs** (Numerik 3)
5. Class 3b: Finite-Elemente-Programmierpraktikum (C++)
6. Class 3c: Goal-oriented error estimation and adaptive finite elements
7. Class 4: Numerical methods for continuum mechanics
8. Special lectures:
   - Class 5a: Numerical methods for nonlinear and coupled problems,
   - Class 5b: Numerical methods for phase-field fracture,
   - Class 5a/b: Numerical methods for coupled variational inequality systems,
   - Class 5c: Scientific computing
   - ...

Contents

1 Main literature and acknowledgements

1.1 Linear PDEs and finite elements

1. C. Johnson 1987; Numerical solution of partial differential equations by the finite element method [82] (in English; original version in Swedish)
2. P. Ciarlet 1987: The finite element method for elliptic problems [35] (in English)
3. R. Rannacher 2017 (in German); Numerik partieller Differentialgleichungen [106]
4. K. Eriksson and D. Estep and P. Hansbo and C. Johnson: Computational Differential Equations 2009 [48] (in English)
5. Ch. Grossmann, H.-G. Roos, M. Stynes; Numerical treatment of partial differential equations [65] (in English; original version in German).
6. T. Hughes 2000; The finite element method [78] (in English)
7. G. F. Carey and J. T. Oden 1984; Finite Elements. Volume III. Compuational Aspects [30] (in English)
8. D. Braess 2007; Finite Elemente [24] (in German)
9. G. Allaire, F. Alouges: Analyse variationnelle des équations aux dérivées partielles [3] (in French)
10. S. Brenner and L.R. Scott 2008; The mathematical theory of finite element methods [26] (in English)
11. W. Hackbusch 1986: Theorie und Numerik elliptischer Differentialgleichungen [68] (in German; also available in English)
12. H.R. Schwarz: Methode der finiten Elemente [120] (in German)
13. G. Allaire: Introduction to mathematical modelling, numerical simulation, and optimization [2] (in English; original version in French),
14. P.G. Ciarlet: Linear and nonlinear functional analysis [36] (in English).

1.2 Nonlinear and coupled PDEs

(the order of the following list is purely arbitrary)

1. G. Duvaut, J.L. Lions: Inequalities in mechanics and physics, Springer, 1976 [43]
2. R. Glowinski; Numerical methods for nonlinear variational problems, 1982
3. Trémolières/Lions/Glowinski; Numerical Analysis of VI, 2011 (original version in French) [130]
4. R. Glowinski; P. Le Tallec; AL and OS in nonlinear mechanics [59]
5. F.-T. Suttmeier; Numerical solution of Variational Inequalities by Adaptive Finite Elements, Vieweg-Teubner, 2008 [123]
6. T. Richter; Fluid-structure interactions: models, analysis, and finite elements, Springer, 2017 [112]
7. J. M. Ortega, W.C. Rheinboldt; Iterative solution of nonlinear equations in several variables, Academic Press, New York, 1970
8. H. Schwetlick; Numerische Lösung nichtlinearer Gleichungen. Verlag der Wissenschaften, Berlin, 1978
9. N. Kikuchi, J.T. Oden; Contact problems in elasticity, SIAM, 1988. [83]
10. S. Bartels; Numerical methods for nonlinear PDEs, 2015 [12]
11. T. Wick; Multiphysics phase-field fracture, de Gruyter, 2020 [141]

1.3 Own materials

For the definition of the 'Classes', we refer to page 3.

1. Class 1: T. Richter, T. Wick; Einführung in die numerische Mathematik - Begriffe, Konzepte und zahlreiche Anwendungsbeispiele, Springer, 2017 [114] (in german)
   https://www.springer.com/fr/book/9783662541777
2. Class 1 (english version; École Polytechnique): S. Amstutz, T. Wick; Refresher course in maths and a project on numerical modeling done in twos. Hannover : Institutionelles Repositorium der Leibniz Universität Hannover, 2021-12-30, 276 S. [5] (in english)
   https://doi.org/10.15488/11629
3. Class 1b: M.C. Steinbach, J.P. Thiele, T. Wick; Algorithmisches Programmieren (Numerische Algorithmen mit C++) Hannover : Institutionelles Repositorium der Leibniz Universität Hannover, 2021, [122] (in german)
   https://doi.org/10.15488/11583
4. Class 2: L. Samolik, T. Wick; Numerische Methoden für gewöhnliche Differentialgleichungen und Eigenwertprobleme, Mitschrift/Skriptum, SS 2019, Leibniz Universität Hannover, 2019 [117] (in german)
   and
   Class 2/3: T. Wick, P. Bastian: PeC3 Spring School on Introduction to Numerical Modeling with Differential Equations. Hannover : Institutionelles Repositorium der Leibniz Universität Hannover, 2019. DOI: https://doi.org/10.15488/6427 [143] (in english)
5. Classes 3, 3b, 3c: These lecture notes (in english)
6. Class 3c: J. Goldenstein, T. Wick; Goal-oriented a posteriori error estimation and adaptive finite elements; Lecture notes WS 19/20, Leibniz Universität Hannover, Feb 2020 (in english)
7. Class 4: T. Wick; Modeling, Discretization, Optimization, and Simulation of Fluid-Structure Interaction; Lecture notes at Heidelberg University, TU Munich, and JKU Linz, 2016, available on https://www-m17.ma.tum.de/Lehrstuhl/LehreSoSe15NMFSIEn (in english)
8. Class 5a: These lecture notes (in english)
9. Class 5b: K. Mang, T. Wick; Numerical methods for variational phase-field fracture problems; Hannover : Institutionelles Repositorium der Leibniz Universität Hannover, 2019 DOI: https://doi.org/10.15488/5129 (in english)
10. Classes 4, 5a, 5b (additional literature): T. Wick; Multiphysics phase-field fracture, de Gruyter, 2020 [141], DOI: http://www.degruyter.com/books/978-3-11-049656-7

1.4 Online materials (WS 20/21) for IntNumPDE (an international open class)

Due to the COVID-19 online lectures, students of mine (Max Schröder and Julian Roth) complemented these lecture notes with additional videos published on youtube:

- PDEs and their classification: https://www.youtube.com/watch?v=afNZj5URZsc&feature=youtu.be
- Finite differences: https://youtu.be/YotrBNLFen0
- Finite elements: https://youtu.be/P4lBRuY7pC4
- Galerkin orthogonality and Céa's lemma: https://www.youtube.com/watch?v=uPt5PIW1Rfo
- Lax-Milgram lemma: https://www.youtube.com/watch?v=w8QjyVug78M

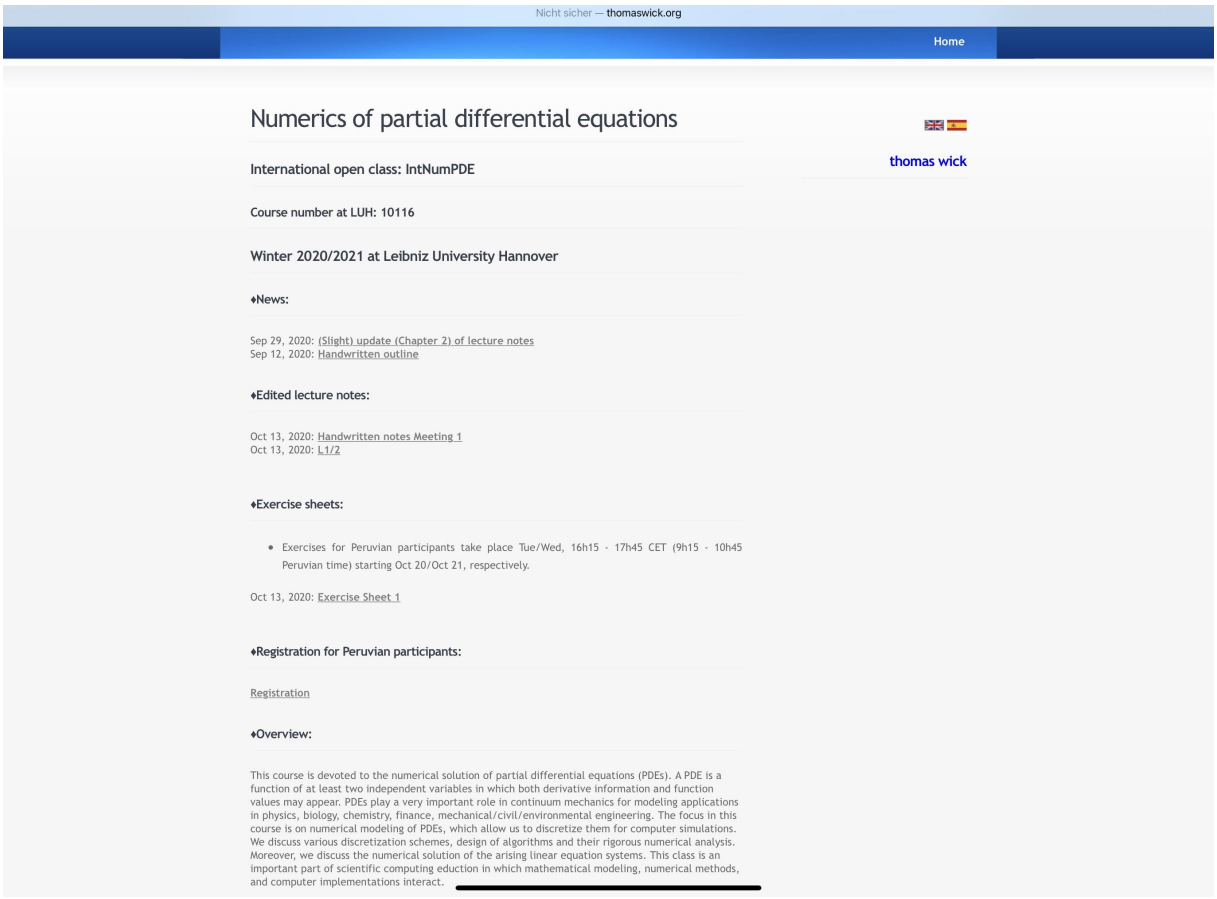
All materials of this online class can be found on:

- Handwritten notes, edited lecture notes, exercise sheets can be found on: http://www.thomaswick.org/num_pde_winter_2020_engl.html

1.5 IntNumPDE: an international open class and computing future lectures

Link to the webpage:

http://www.thomaswick.org/num_pde_winter_2020_engl.html



1.6 Teaching funding acknowledgements (WS 20/21)

Due to the COVID-19 global crisis, financial support for improving teaching lecture materials (such as these notes and the previously mentioned videos) is gratefully acknowledged:

- DAAD (German Academic Exchange Service): PeCCC (Peruvian Competence Center of Scientific Computing), Stärkung des Wissenschaftlichen Rechnens in der Lehre in Peru, https://www.pec3.org/
- Leibniz University Hannover: International Office, South America support administrated from Ms. Rhina Colunge-Peters
- Leibniz University Hannover: Fonds für Internationalisierung

2 Motivation

These lecture notes are devoted to **the numerical solution of partial differential equations (PDEs)**. A PDE is a function of at least two independent variables in which both derivative information and function values may appear. PDEs play important roles in continuum mechanics for modeling applications in physics, biology, chemistry, finance, mechanical/civil/environmental engineering. The focus in this course is on numerical modeling of PDEs, which allow us to discretize them for computer simulations. We discuss various discretization schemes, design of algorithms and their rigorous numerical analysis. Moreover, we discuss the numerical solution of the arising linear equation systems. This class is an important part of scientific computing eduction in which mathematical modeling, numerical methods, and computer implementations interact. In these notes, not only classical topics for linear PDEs, methods for nonlinear PDEs, and coupled problems are addressed, but we do also dive into current state-of-the-art techniques.

2.1 Scientific computing (english) / Wissenschaftliches Rechnen (german)

Developing and analyzing algorithms for solving PDEs with a computer is a part of **numerical methods**, which itself is a part of **scientific computing**. Scientific computing comprises three main fields:

1. **Mathematical modeling** and analysis of physical, biological, chemical, economical, financial processes, and so forth;
2. Development of reliable and efficient **numerical methods** and algorithms and their analysis;
3. **Research software development**: Implementation of the derived algorithms

All these steps work in a **feed-back manner** and the different subtasks interact with each other. It is in fact the third above aspect, namely *software and computers*, who helped to establish this third category of science. Thus, a new branch of mathematics, *computational science*, has been established. This kind of mathematics may become experimental like experiments in physics/chemistry/biology.

A key task is the design and analysis of algorithms:

Definition 2.1 (Algorithm)**.** *An algorithm is an instruction for a schematic solution of a mathematical problem statement. The main purpose of an algorithm is to formulate a scheme that can be implemented into a computer to carry out so-called numerical simulations. Direct schemes solve the given problem up to round-off errors (for instance Gaussian elimination). Iterative schemes approximate the solution up to a certain accuracy (for instance Richardson iteration for solving linear equation systems, or fixed-point iterations). Algorithms differ in terms of* ***accuracy, robustness****, and* ***efficiency****.*

An important feedback task is to analyse (rigorously or computationally) these algorithms in order to detect shortcomings and suggest improvements. These can be nowadays on the algorithmic side (classical numerical analysis with convergence proofs) or the computational side (analysis of different discretization levels, parameter variations by just running the code again) or coding improvements (re-organization of, e.g., for-loops in a finite element program), or hardware-specific aspects (e.g., CPU computing, GPU computing).

2.1.1 Addressing new problems

Computational science allows us to investigate research fields that have partially not been addressable in the past. Why? On the one hand experiments are often too expensive, too far away (Mars, Moon, astronomy in general), the scales are too small (nano-scale for example); or experiments are simply too dangerous. On the other hand, mathematical theory or the explicit solution of an (ambitious) engineering problem in an analytical manner is often impossible!

Nowadays, **nonstationary, nonlinear, coupled, PDE systems** are addressed in which two or more PDEs interact. Sometimes, these systems are subject to inequality constraints resulting into **CVIS** (coupled variational inequality systems) [141]. Mathematics helps to formalise and structure such systems. Hereupon, established algorithms for the numerical discretization and numerical solution for known parts of the given system shall be further extended for new problems and new applications.

2.2 Differential equations

Let us first define the meaning of a differential equation:

Definition 2.2. *A differential equation is a mathematical equation that relates a function with its derivatives such that the solution satisfies both the function and the derivatives.*

Differential equations can be split into two classes:

Definition 2.3 (Ordinary differential equation (ODE))**.** *An ordinary differential equation (ODE) is an equation (or equation system) involving an unknown function of one independent variable and certain of its derivatives.*

Definition 2.4 (Partial differential equation (PDE))**.** *A partial differential equation (PDE) is an equation (or equation system) involving an unknown function of two or more variables and certain of its partial derivatives.*

Such differential equations are posed in function spaces and analyzed therein. For the following, let us denote:

- V function space of the (possibly unknown) exact solution u. We write $u \in V$.
- $\tilde{V}$ function space of approximate solutions $\tilde{u}$. We write $\tilde{u} \in \tilde{V}$.

Numerical discretizations in space and time introduce parameters h and k in order to compute the approximate solution, i.e., $\tilde{u} := u_{hk}$. We then aim for that the **discretization error** behaves as

$$\|u - u_{kh}\| \to 0 \quad \text{for } h \to 0,\ k \to 0.$$

Further errors may come into play as we will see after the next subsection. Moreover, the choice of good norms is important to determine the **accuracy.**

2.3 Guiding questions in differential equations

1. What type of equations are we dealing with?
2. Are we dealing with a single PDE, coupled problems, nonlinear PDEs?
3. What are the physical conservation properties?
4. Are theoretical results known? Can the equations be related to (simplified) known PDEs?
5. What kind of discretization scheme(s) shall we use?
6. How do we design algorithms to compute the discrete solution u_h?
7. Can we proof that these algorithms really work?
8. Are they robust (stable with respect to parameter variations), accurate, and efficient?
9. Can we construct physics-based algorithms that maintain as best as possible conservation laws; in particular when several equations interact (couple)?
10. How far is u_h away from u in a certain (error) norm? Hint: comparing color figures gives a first impression, but is not science!
11. The discretized systems (to obtain u_h) are often large with a huge number of unknowns: how do we solve these linear equation systems?
12. What is the computational cost?
13. How can we achieve more efficient algorithms? Hint: adaptivity, parallel computing, model order reduction.

14. Are analytical solutions or experimental results available to compare with?
15. How can we check that the solution is correct?
16. How do we present numerical results: figures, graphs, tables?
17. What parts of the solution are of interest? The entire solution? Only certain goal functionals?
18. What is my personal task or goal: theory, numerical analysis or implementation? For the latter, what software is available on the market?

2.4 Errors

In order to realize algorithms, we go ahead and implement them in a programming language or software (for instance Matlab/octave, python, fortran, C++) using a computer or cluster.

From the cycle over mathematical modeling, over the design of algorithms, up to their implementation, we face several error sources:

1. The set of numbers that can be represented in a computer is finite therefore a numerical calculation is limited by machine precision, which results in **round-off errors**.
2. The memory of a computer (or cluster) is finite and thus functions and equations can only be represented through approximations. Thus, continuous information has to be represented through discrete information, which results in investigating so-called **discretization errors** or more general **approximation error**.
3. **Interpolation errors** appear when complicated functions are interpolated to simpler functions.
4. All further simplifications of a numerical algorithm (in order to solve the discrete problem), with the final goal to reduce the computational time, result into **iteration / truncation errors**. One example is the stopping criterion, which decides after how many steps an iteration is stopped. These can be further divided into **linear** and **nonlinear** iteration errors.
5. **Regularization errors** appear when ill-posed models are adapted such that they become well-posed.
6. **Homogenization errors** arise when full-scale models are reduced (homogenized) such that they are computationally simpler to solve.
7. **Implementation errors**, better known as **bugs**, appear when mistakes are made implementing algorithms into a software. These errors can simply cause the program to abort. Then it is more or less simple to find the place with the help of a debugger. But there are also subtle errors, which cannot be easily found when the program runs nonetheless, but showing strange results or behaviors.
8. **Model errors**: In order to make a 'quick guess' of a possible solution and to start the development of an algorithm aimed at addressing at a later stage a difficult problem, often complicated (nonlinear) differential equations are reduced to simple (in most cases linear) versions, which results in the so-called **model error**.
9. **Data errors and uncertainties**: the data (e.g., input data, boundary conditions, parameters) are obtained from experimental data and may be inaccurate themselves.
10. **Inaccurate experimental setups** that yield wrong reference solutions u to which numerical solutions $\tilde{u}$ should be compared with.

It it very important to understand and to accept that we **never can avoid** all these errors. The important aspect is to **control** these errors and to provide answers if these errors are too big to influence the interpretation of numerical simulations or if they can be assumed to be negliable. A big branch of numerical mathematics is to derive **error estimates** that allow us to predict quantitative information about the arising errors.

One objective is to clearify whether different error parts are rather of theoretical value or whether they are computable. Also, we need to study which error contribution dominates.

2.4.1 Illustration of three errors (experimental, theory, numerical) with the clothesline problem

You need to understand a bit of French for the words, but the figure might be self-explaining.

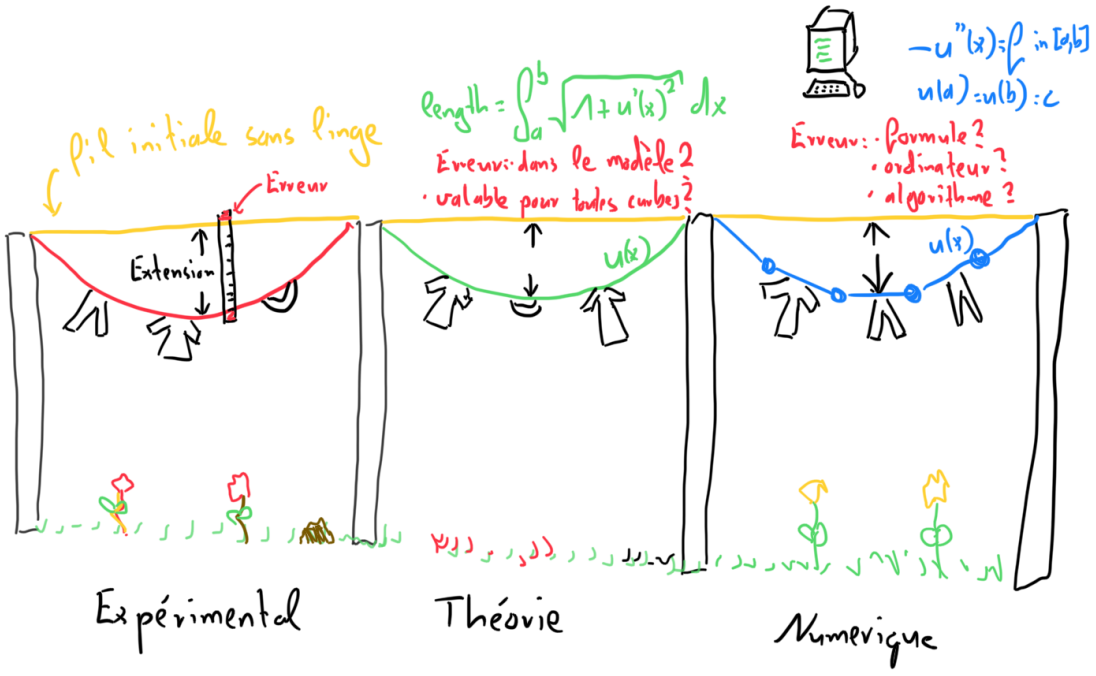


Figure 1: Errors in experiments, theory and numerical modeling.

2.4.2 A priori and a posteriori error estimates

Error estimates can be classified into two categories:

- **a priori estimates** include the (unknown) exact solution u, such that $\eta := \eta(u)$, and yield qualitative convergence rates for asymptotic limits. They can be derived before (thus a priori) the approximation is known.
- **a posteriori error estimates** are of the form $\eta := \eta(\tilde{u})$ explicitly employ the approximation $\tilde{u}$ and therefore yield quantitative information with computable majorants (i.e., bounds) and can be further utilized to design **adaptive schemes** (see Chapter 9).

2.5 Accuracy

According to the ISO norm 5725-1:1994, **'accuracy is the closeness of an agreement between a test result and an accepted reference value'**. The justification of the reference value can be an issue of itself. Closely related is the **precision**, which measures how closely two or more measurements agree with each other. This is linked to reproducibility. Traditionally, precision rather comes from experimental observations, but since numerical simulations are also 'experimental' in view of various parameters, precision may play a role here as well.

Accuracy and precision can be quantified by employing error estimates. Mathematically, we need to frame our problem statements in appropriate spaces that allow us to measure distances (i.e., the distance between the test result and the reference value). A metric space is a set X with a metric on it. This metric associates with any pair of elements (i.e., points), say $a, b \in X$, a distance. The concept of a distance is more general than that of a norm.

Definition 2.5 (Metric space)**.** *A metric space is a pair (X, d) where X is a set and d is a metric (distance function) on X. The function d is defined on $X \times X$, where $\times$ denotes the Cartesian product of sets such that $X \times X$ is the set of all ordered pairs of elements of X. For all $x, y, z \in X$, it holds*

1. *d is real-valued, finite and nonnegative: $d \in [0, \infty)$*
2. *$d(x, y) = 0$ if and only if $x = y$*
3. *$d(x, y) = d(y, x)$*
4. *$d(x, y) \leq d(x, z) + d(z, y)$*

Example 2.6. *On the real line $\mathbb{R}$ the usual metric is defined by*

$$d(x, y) = |x - y|.$$

The Euclidian plane $\mathbb{R}^2$ and the Euclidian space $\mathbb{R}^3$ are natural extensions. As another example, we obtain the function space $C[a, b]$ of continuous functions defined over the closed interval $[a, b]$. Further examples can be found in any calculus (analysis) or functional analysis textbook.

The specification of a metric from the two-dimensional plane or our three-dimensional usual space in which we live, is a so-called norm. Thus, **norms provide distance measures in more general spaces**. This leads to **normed spaces.** Such norms can now be employed to study the accuracy of mathematical models and their numerical approximations.

Definition 2.7 (Normed space)**.** *A normed space X is a vector space with a norm. A norm on a real or complex vector space X is a real-valued function on X whose value at $x \in X$ is denoted by $\|x\|$ and which has the following properties:*

1. *$\|x\| \geq 0$*
2. *$\|x\| = 0 \Leftrightarrow x = 0$*
3. *$\|\alpha x\| = |\alpha| \|x\|$*
4. *$\|x + y\| \leq \|x\| + \|y\|$*

where $x, y \in X$ and α is any scalar from the underlying field $\mathbb{R}$ or $\mathbb{C}$.

Definition 2.8 (Norm vs. metric)**.** *A norm on X induces a metric d on X, which is defined by*

$$d(x, y) = \|x - y\|.$$

With the help of these definitions it is easy to infer that indeed a norm induces a metric. Therefore, normed spaces are in particular metric spaces. It holds

Proposition 2.9. *The norm is continuous, that is $x \mapsto \|x\|$ is a continuous mapping of $(X, \|\cdot\|)$ into the field $\mathbb{R}$.*

Distances, induced or not by a norm, can now be employed to study the accuracy of mathematical models and their numerical approximations.

2.5.1 Consequences for numerical mathematics

Closing the loop to Section 2.2, one key task is often to design appropriate norms for measuring the distance between a numerical approximation u_h and the continuous (exact) solution u. These choices depend on the geometry, regularity properities of coefficients, functions, and right hand sides. Such norms give quantitative information about the (approximation) error for deciding when an approximation (here discretization) is sufficiently close to our sought solution:

$$\|u - u_{hk}\| \to 0 \quad \text{for } h \to 0,\ k \to 0.$$

As another example, iteration error measurements allow us to decide when numerical algorithms can be stopped:

$$\|u - u_l\| \to 0 \quad (l \to \infty),$$

where l indicates the iteration index. Very often, a mixture of both (and more situations) appear simultaneously.

A natural question is whether different norms yield similar answers. Here, one must distinguish between finite-dimensional vector spaces and infinite-dimensional spaces. The latter results into the branch of functional analysis. Therein, norms are in general not equivalent. Consequently, it does matter which norm is adopted. In finite-dimensional spaces, it can be shown, however, that norms are indeed equivalent, but the involved constants might influence the results.

2.6 Concepts in numerical mathematics

2.6.1 Definitions

In introductory classes to numerical methods, we deal with concepts that are very characteristic for numerical modeling. In [114], we summarized them into seven points, which will frequently encounter us in the forthcoming chapters:

1. **Approximation**: since analytical solutions are not possible to achieve as we just learned in the previous section, solutions are obtained by **numerical approximations**.

2. **Convergence**: is a qualitative expression that tells us when members a_n of a sequence $(a_n)_{n\in\mathbb{N}}$ are sufficiently close to a limit a. In numerical mathematics this limit is often the solution that we are looking for.

3. **Order of convergence**: While in analysis, we are often interested in the convergence itself, in numerical mathematics we must pay attention how long it takes until a numerical solution has sufficient accuracy. The longer a simulation takes, the more time and more energy (electricity to run the computer, air conditioning of servers, etc.) are consumed. Therefore, we are heavily interested in developing fast algorithms. In order to judge whether an algorithm is fast or not we have to determine the order of convergence.

4. **Errors**: Numerical mathematics can be considered as the branch 'mathematics of errors' as we discussed in detail in Section 2.4.

5. **Error estimation**: This is one of the biggest and most classical branches in numerical mathematics. We need to derive error formulae to judge the outcome of our numerical simulations and to measure the difference of the numerical solution $\tilde{u}$ and the (unknown) exact solution u in appropriate norms. A crucial aspect is to find out which error contributions dominate the total error. Say we have

   $$e_{total} = e_{disc} + e_{model}$$

   Just as an example, let us assume $e_{disc} \gg e_{model}$, then we can work with

   $$e_{total} \approx e_{disc}.$$

   It in the very heart of numerics to derive such error estimates and to decide which terms dominate. It may happen that in lucky cases very complicated terms do not need to be implemented because their error contribution is negligable. In a book review [142], I review a work from Repin and Sauter (2020) [109] in which I explain the purpose of error estimates for the general mathematical community and not only for numerical specialists.

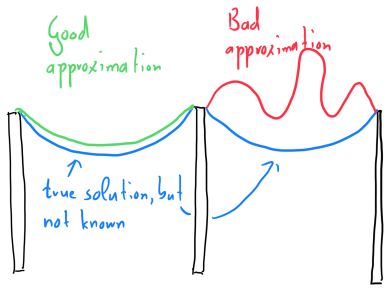
6. **Efficiency**: In general we can say, the higher the convergence order of an algorithm is, the more efficient the algorithm is. Therefore, we obtain faster the numerical solution to a given problem. But numerical efficiency is not automatically related to resource-effective computing. For instance, developing a parallel code using MPI (message passing interface), hardware-optimization (CPU,GPU), software optimizations (ordering in some optimal way for-loops, arithmetic evaluations, etc.) can further reduce computational costs.

7. **Stability**: Lastly, the robustness of algorithms and implementations with respect to parameter (model, material, numerical) variations, boundary conditions, initial conditions, uncertainties must be studied. Stability relates in the broadest sense to the third condition of Hadamard defined in Section 2.8.

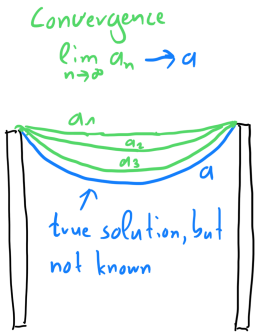
2.6.2 Examples

We illustrate the previous concepts with the help of some examples.

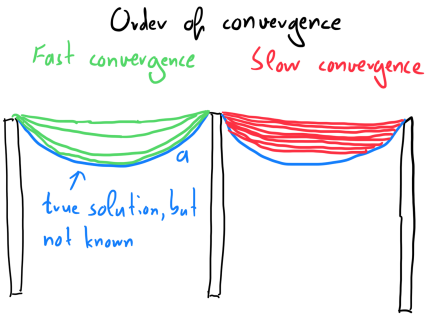
1. **Approximation**: Two approximations of the clothesline problem:



2. **Convergence**: Converging approximations of the clothesline problem:



3. **Order of convergence**: Two different speeds of convergence:



4. **Errors**: We first refer the reader to Section 2.7 for a comparison of experimental, theoretical and numerical situations. Second, we sharpen the sense of the influence of different errors. Not all errors

are equally important and sometimes, one might try to 'optimize' an error, which has no significant influence. Let's see this in more detail. Let the errors $e_{Model}, e_{Numerics}, e_{Software}$ enter. The total error is defined as

$$e_{Total} = e_{Model} + e_{Numerics} + e_{Software}$$

Let us assume that we have the numbers $e_{Model} = 1000, e_{Numerics} = 0.001, e_{Software} = 4$, the total error is then given by

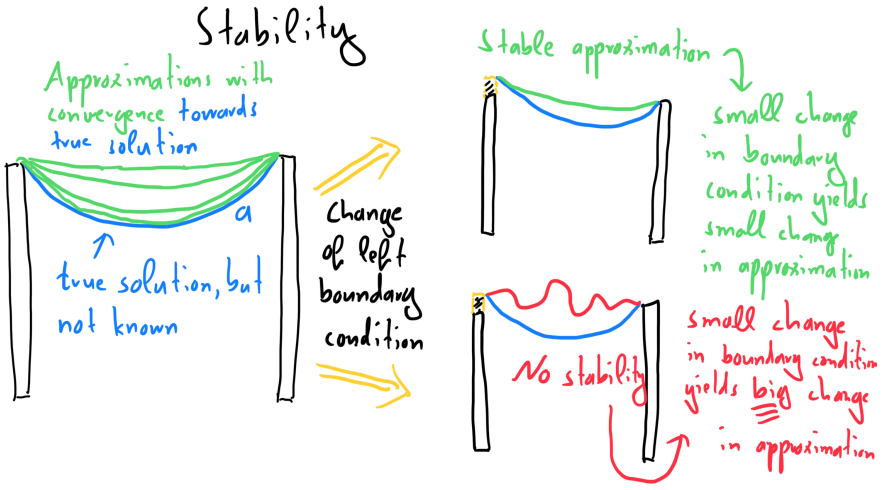
$$e_{Total} = 1000 + 0.001 + 4 = 1004.001.$$

Which error dominates? It is clearly $e_{Model} = 1000$. The relative influence is $e_{Model}/e_{Total} = 0.996$. So, the other two error sources are negligible and would not need further attention in this specific example.

5. **Error estimation**: Error estimation is the process to obtain the concrete numbers $1000, 0.001, 4$ in the previous example. Error estimates can be classified into two categories:
   - **a priori estimates** include the (unknown) exact solution u, such that $\eta := \eta(u)$, and yield qualitative convergence rates for asymptotic limits. They can be derived before (thus a priori) the approximation is known.
   - **a posteriori error estimates** are of the form $\eta := \eta(\tilde{u})$ explicitly employ the approximation $\tilde{u}$ and therefore yield quantitative information with computable majorants (i.e., bounds) and can be further utilized to design **adaptive schemes**.

6. **Efficiency**: is more or less self-explaining. A first answer is to look at CPU or wall time: how many seconds, minutes, weeks, months does a program need to terminate and yield a result? A second answer is to study 'iteration numbers' or arithmetic operations. The latter are often given in terms of the big O notation. Having a linear equation system $Ax = b$ with $A \in \mathbb{R}^{n \times n}$ and $O(n^3)$ complexity means that we need n^3 (cubic in unknowns n) arithmetic operations to calculate the result. For instance, for $n = 100$, we need around $1\,000\,000$ operations. Having another algorithm (yielding the same result of $Ax = b$) with only $O(n)$ operations, means that we only need around 100 operations, which is a great difference. The development of efficient solvers for large linear equations systems is consequently a big branch in numerics and scientific computing.

7. **Stability**: We finally come back to the clothesline problem and change a bit the left boundary condition:



2.7 Illustration of physical oscillations versus numerical instabilities

To illustrate the numerical concept No. 7, we study an example (taken from [135]) of periodic flow interacting with an elastic beam (fluid-structure interaction).

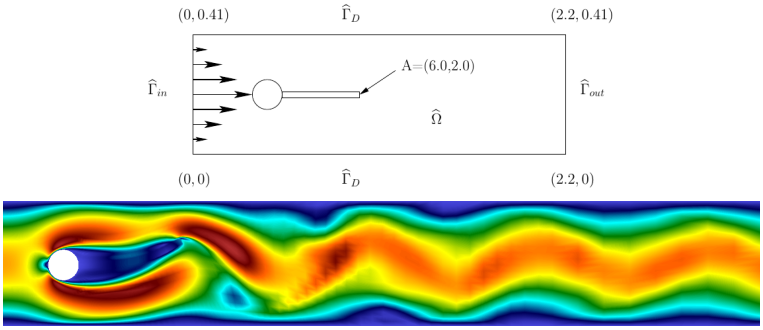
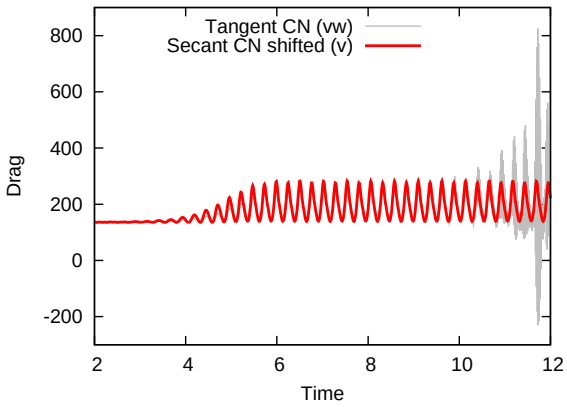


Figure 2: Fluid flow (Navier-Stokes) interacts with an elastic beam. Due to a non-symmetry of the cylinder, the beam starts oscillating. These oscillations are physical!

- Observe the tip of the elastic beam!

→ Physical oscillations! Shown in red color for a 'good' numerical scheme.



- The grey numerical scheme exhibits at some time around $t \approx 10$ micro-oscillations which are due to numerical instabilites. Finally the grey numerical scheme has a blow-up and yields garbage solutions.

2.8 Well-posedness (in the sense of Hadamard 1923)

The concept of well-posedness is very general and in fact very simple.

Definition 2.10. *Let $A : X \to Y$ be a mapping and X, Y topological spaces. Then, the problem*

$$Ax = y \tag{1}$$

is well posed if

1. *for each $y \in Y$, Problem (1) has a solution x,*
2. *the solution x is unique,*
3. *the solution x depends continuously on the problem data.*

The first condition is immediately clear. The second condition is also obvious but often difficult to meet - and in fact many physical processes do not have unique solutions. The last condition says if a variation of the input data (right hand side, boundary values, initial conditions) vary only a little bit, then also the (unique) solution should only vary a bit.

Remark 2.11. *Problems in which one of the three conditions is violated are **ill-posed**.*

Example 2.12 (Numerical differentiation)**.** *In view of the later chapters and that we shall deal with derivatives throughout these notes, we consider the difference quotient $D_h g$ as approximation of the derivative g' of a function g. We know from introduction to numerics:*

$$\begin{aligned}
D_h g(x) &= \frac{g(x+h) - g(x)}{h}, \quad 0 \le x \le h/2, \\
D_h g(x) &= \frac{g(x+h/2) - g(x-h/2)}{h}, \quad h/2 < x < 1 - h/2, \\
D_h g(x) &= \frac{g(x) - g(x-h)}{h}, \quad 1 - h/2 \le x \le 1,
\end{aligned}$$

with a step size h. We know that

$$\begin{aligned}
|g'(x) - D_h g(x)| &= O(h^2) \quad \text{for the central difference quotient} \\
|g'(x) - D_h g(x)| &= O(h) \quad \text{for the forward/backward difference quotients},
\end{aligned}$$

when g is three times continuously differentiable. When g is only of class $C^2[0,1]$ or when we only have disturbed data g_ε (e.g., data or model or round-off errors) with $\|g - g_\varepsilon\| \le \varepsilon$, then we have for the data error

$$|D_h(g_\varepsilon - g)(x)| \le \frac{2\varepsilon}{h}, \quad 0 \le x \le 1.$$

The total error is composed by the data error $D_h(g_\varepsilon - g)$ and the discretization error $g' - D_h g$:

$$D_h g_\varepsilon - g' = D_h(g_\varepsilon - g) + D_h g - g'$$

yielding the estimate

$$\|D_h g_\varepsilon - g'\| \le \frac{2\varepsilon}{h} + \frac{1}{2}\|g''\|_{C^\infty} h.$$

*This means that for noisy data, i.e., $\varepsilon \neq 0$, the total error cannot become arbitrarily small as we normally hope to wish. **This problem is ill-posed - specifically for $h \ll \varepsilon$.** The optimal step size is*

$$h_\varepsilon = \frac{2}{\|g''\|_{C^\infty}^{1/2}} \varepsilon^{1/2},$$

if $g'' \neq 0$. In conclusion the total error is of order $O(\varepsilon^{1/2})$.

Example 2.13 (An ill-posed elliptic equation from Hadamard)**.** *Let $\Omega := (-\infty, \infty) \times (0, \delta)$. We consider the Laplace equation (defined in Chapter 4). We define the following boundary value problem:*

$$\begin{aligned} \Delta u &= 0 \quad \text{in } \Omega \\ u(x,0) &= \varphi(x) \\ \partial_y u(x,0) &= 0 \end{aligned}$$

with the boundary function $\varphi(x) = \frac{\cos(x)}{n}$. To this problem, we can compute the exact solution

$$u(x,y) = \frac{\cos(nx)\cosh(ny)}{n}.$$

For $n \to \infty$ we easily see the limits:

$$\begin{aligned} \lim_{n\to\infty} \varphi(x) &\to 0 \\ \lim_{n\to\infty} u(x,y) &\to \infty \end{aligned}$$

for the general cases when we do not sit in a period $x = \pi/2$ of $\cos(nx) = 0$. Consequently, ***small boundary data*** *cause large solution deflections, which is therefore an ill-posed problem.*

2.9 Numerical methods for solving PDEs

There exist various methods for solving PDEs numerically:

- Finite differences: the derivatives in the differential equation will be replaced by difference quotients. As they belong to the most simple methods and are still used frequently in engineering and industry, we will provide a short introduction on them as well.
- Finite volume methods: they are based on local conservation properties of the underlying PDE.
- Variational methods such as Galerkin finite elements (FEM).
- Boundary element methods (BEM).
- Isogeometric analysis (IGA) [79].
- Spectral methods.
- Meshless methods.

2.10 Chapter summary and outlook

In this chapter, we recapitulated guiding questions and numerical concepts. We also defined in a brief way a PDE. In the next chapter, we shall introduce the notation used throughout these lecture notes.

3 Notation

In this chapter, we collect most parts of the notation and some important results from linear algebra, analysis, vector and tensor analysis, necessary for numerical mathematics.

3.1 Domains

We consider open, bounded domains $\Omega \subset \mathbb{R}^d$ where $d = 1, 2, 3$ is the dimension. The boundary is denoted by $\partial\Omega$. The outer **normal vector** with respect to (w.r.t.) to $\partial\Omega$ is n. We assume that Ω is sufficiently smooth (i.e., a Lipschitz domain or domain with Lipschitz boundary) such that the normal n can be defined. What also works for most of our theory are convex, polyhedral domains with finite corners. For specific definitions of nonsmooth domains, we refer to the literature, e.g., [63].

3.2 Independent variables

A point in $\mathbb{R}^d$ is denoted by

$$x = (x_1, \dots, x_d).$$

The variable for 'time' is denoted by t. The euclidian scalar product is denoted by $(x, y) = x \cdot y = \sum_{i=1}^{d} x_i y_i$.

3.3 Function, vector and tensor notation

We start with

Definition 3.1. *A real-valued or complex-valued function (of one variable) on a set X is a mapping f, which associates in a unique fashion to each $x \in X$ a number $f(x)$. We write $f : X \to \mathbb{C}$ or $f : X \to \mathbb{R}$ and $x \mapsto f(x)$. The set X is the definition space and $f(X) := \{f(x) \in \mathbb{C} | x \in X\}$ is the image set. The graph of f is the set*

$$G(f) := \{(x, f(x)) | x \in X\} \subset X \times \mathbb{C}.$$

The variable x is the 'independent' variable or also called the argument. All x values form the set of definition of $f(x)$. The variable y is called 'dependent'. All y values form the image space of $f(x)$.

In these notes, functions are often denoted by

$$u := u(x)$$

if they only depend on the spatial variable $x = (x_1, \dots, x_d)$. If they depend on time and space, they are denoted by

$$u = u(t, x).$$

Usually in physics or engineering vector-valued and tensor-valued quantities are denoted in bold font size or with the help of arrows. Unfortunately in mathematics, this notation is only sometimes adopted. We continue this crime and do not distinguish scalar, vector, and tensor-valued functions. Thus for points in $\mathbb{R}^3$ we write:

$$x := (x, y, z) = \mathbf{x} = \vec{x}.$$

Similar for functions from a space $u : \mathbb{R}^3 \supseteq U \to \mathbb{R}^3$:

$$u := (u_x, u_y, u_z) = \mathbf{u} = \vec{u}.$$

And also similar for tensor-valued functions (which often have a bar or two bars under the tensor quantity) as for example the Cauchy stress tensor $\sigma_f \in \mathbb{R}^{3\times3}$ of a fluid:

$$\sigma_f := \underline{\sigma}_f = \begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{pmatrix}.$$

3.4 Partial derivatives

We frequently use:

$$\frac{\partial u}{\partial x} = \partial_x u$$

and

$$\frac{\partial u}{\partial t} = \partial_t u$$

and

$$\frac{\partial^2 u}{\partial t \partial t} = \partial_t^2 u$$

and

$$\frac{\partial^2 u}{\partial x \partial y} = \partial_{xy} u$$

3.5 Chain rule

Let the functions $g : (a, b) \to \mathbb{R}^m$ and $f : \mathbb{R}^m \to \mathbb{R}$ and its composition $h = f(g) \in \mathbb{R}$ be given and specifically $g := (t, x) := (t, x_2, x_3, \ldots, x_m)$:

$$\begin{aligned} D_t h(x) &= D_t f(g(x)) = D_t f(t, x) = D_t f(t, x_2, \ldots, x_k) \\ &= \sum_{k=1}^{m} \partial_k f(g(x)) \cdot \partial_t g_k \\ &= \sum_{k=1}^{m} \partial_k f(t, x_2, \ldots, x_m) \cdot \partial_t x_k, \quad \text{where } x_1 := t \\ &= \partial_t f \cdot \partial_t t + \sum_{k=2}^{m} \partial_k f(t, x_2, \ldots, x_k) \cdot \partial_t x_k \\ &= \partial_t f + \nabla f \cdot (\partial_t x_2, \cdots, \partial_t x_m)^T. \end{aligned}$$

For instance $m = 4$ means that we deal with a four-dimensional continuum (t, x, y, z).

Remark 3.2. *See also [86][page 54 and page 93] for definitions of the chain rule.*

3.6 Multiindex notation

For a general description of ODEs and PDEs the multiindex notation is commonly used.

- A multiindex is a vector $\alpha = (\alpha_1, \ldots, \alpha_n)$, where each component $\alpha_i \in \mathbb{N}_0$. The order is

$$|\alpha| = \alpha_1 + \ldots + \alpha_n.$$

- For a given multiindex we define the partial derivative:

$$D^\alpha u(x) := \partial_{x_1}^{\alpha_1} \cdots \partial_{x_n}^{\alpha_n} u$$

- If $k \in \mathbb{N}_0$, we define the set of all partial derivatives of order k:

$$D^k u(x) := \{D^\alpha u(x) : |\alpha| = k\}.$$

Example 3.3. *Let the problem dimension $n = 3$. Then, $\alpha = (\alpha_1, \alpha_2, \alpha_3)$. For instance, let $\alpha = (2, 0, 1)$. Then $|\alpha| = 3$ and $D^\alpha u(x) = \partial_x^2 \partial_z^1 u(x)$.*

3.7 Gradient, divergence, trace, Laplace, rotation/curl

Well-known in physics, it is convenient to work with the **nabla-operator** to define derivative expressions. The gradient of a single-valued function $v : \mathbb{R}^n \to \mathbb{R}$ reads:

$$\nabla v = \begin{pmatrix} \partial_1 v \\ \vdots \\ \partial_n v \end{pmatrix}.$$

The gradient of a vector-valued function $v : \mathbb{R}^n \to \mathbb{R}^m$ is called **Jacobian matrix** and reads:

$$\nabla v = \begin{pmatrix} \partial_1 v_1 & \dots & \partial_n v_1 \\ \vdots & & \vdots \\ \partial_1 v_m & \dots & \partial_n v_m \end{pmatrix}.$$

The divergence is defined for vector-valued functions $v : \mathbb{R}^n \to \mathbb{R}^n$:

$$\text{div } v := \nabla \cdot v := \nabla \cdot \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} = \sum_{k=1}^{n} \partial_k v_k.$$

The divergence for a tensor $\sigma \in \mathbb{R}^{n \times n}$ is defined as:

$$\nabla \cdot \sigma = \left(\sum_{j=1}^{n} \frac{\partial \sigma_{ij}}{\partial x_j}\right)_{1 \le i \le n}.$$

The trace of a matrix $A \in \mathbb{R}^{n \times n}$ is defined as

$$tr(A) = \sum_{i=1}^{n} a_{ii}.$$

Definition 3.4 (Laplace operator)**.** *The Laplace operator of a two-times continuously differentiable scalar-valued function $u : \mathbb{R}^n \to \mathbb{R}$ is defined as*

$$\Delta u = \sum_{k=1}^{n} \partial_{kk} u.$$

Definition 3.5. *For a vector-valued function $u : \mathbb{R}^n \to \mathbb{R}^m$, we define the Laplace operator component-wise as*

$$\Delta u = \Delta \begin{pmatrix} u_1 \\ \vdots \\ u_m \end{pmatrix} = \begin{pmatrix} \sum_{k=1}^{n} \partial_{kk} u_1 \\ \vdots \\ \sum_{k=1}^{n} \partial_{kk} u_m \end{pmatrix}.$$

Let us also introduce the **cross product** of two vectors $u, v \in \mathbb{R}^3$:

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} \times \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = \begin{pmatrix} u_2 v_3 - u_3 v_2 \\ u_3 v_1 - u_1 v_3 \\ u_1 v_2 - u_2 v_1 \end{pmatrix}.$$

With the help of the cross product, we can define the **rotation** or **curl** :

$$\text{rot} v = \nabla \times v = \begin{pmatrix} \partial_x \\ \partial_y \\ \partial_z \end{pmatrix} \times \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = \begin{pmatrix} \partial_y v_3 - \partial_z v_2 \\ \partial_z v_1 - \partial_x v_3 \\ \partial_x v_2 - \partial_y v_1 \end{pmatrix}.$$

3.8 Some properties of matrices

3.8.1 Eigenvalues

The complex or real number λ is an eigenvalue of the square matrix A if there is a vector $x \in \mathbb{C}^n, x \neq 0$, such that the eigenvalue equation holds:

$$Ax = \lambda x.$$

The vector x is called eigenvector corresponding to the eigenvalue λ. From linear algebra arguments λ is an eigenvalue of A if and only if

$$\det(A - \lambda I) = 0,$$

which is the so-called characteristic equation for A. The left hand side is a polynomial of degree n and called characteristic polynomial of A.

3.8.2 Positive definite, eigenvalues and more

Let $Q : \mathbb{R}^n \to \mathbb{R}$ be a quadratic form with $Q(x) = x^T Ax$. The quadratic form (and also its representing matrix A) have certain names according to their properties:

- Q is positive definite, when $Q(x) > 0$ for all $x \neq 0$
- Q is positive semi-definite, when $Q(x) \geq 0$ for all x
- Q is negative definite, when $Q(x) < 0$ for all $x \neq 0$
- Q is negative semi-definite, when $Q(x) \leq 0$ for all x
- Q is indefinite, when Q has positive and negative values.

The relation to the eigenvalues is:

- $Q > 0$ if and only if all eigenvalues $\lambda_i > 0$
- $Q \geq 0$ if and only if all eigenvalues $\lambda_i \geq 0$
- $Q < 0$ if and only if all eigenvalues $\lambda_i < 0$
- $Q \geq 0$ if and only if all eigenvalues $\lambda_i \leq 0$
- Q indefinite if and only if eigenvalues $\lambda_i < 0$ and $\lambda_i > 0$ arise.

3.8.3 Invariants

The principal invariants of a matrix A are the coefficients of the characteristic polynomial $\det(A - \lambda I)$. A matrix $A \in \mathbb{R}^{3\times 3}$ has three principal invariants; namely $i_A = (i_1(A), i_2(A), i_3(A))$ with

$$\det\,(\lambda I - A) = \lambda^3 - i_1(A)\lambda^2 + i_2(A)\lambda - i_3(A).$$

Let $\lambda_1, \lambda_2, \lambda_3$ be the eigenvalues of A. Then we have

$$\begin{aligned} i_1(A) &= \text{tr}\,(A) = \lambda_1 + \lambda_2 + \lambda_3, \\ i_2(A) &= \frac{1}{2}\left[(\text{tr}\,A)^2 - \text{tr}\,(A)^2\right] = \lambda_1\lambda_2 + \lambda_1\lambda_3 + \lambda_2\lambda_3, \\ i_3(A) &= \det\,(A) = \lambda_1\lambda_2\lambda_3. \end{aligned}$$

Remark 3.6. *If two different matrices have the same principal invariants, they also have the same eigenvalues.*

Remark 3.7 (Cayley-Hamilton)**.** *Every second-order tensor (i.e., a matrix) satisfies its own characteristic equation:*

$$A^3 - i_1A^2 + i_2A - i_3I = 0.$$

3.9 Vector spaces

Let $\mathbb{K} = \mathbb{R}$. In fact, $\mathbb{K} = \mathbb{C}$ would work as well and any general field. But we restrict our attention in the entire lecture notes to real numbers $\mathbb{R}$.

Definition 3.8 (Vector space)**.** *A **vector space** or **linear space** over a field $\mathbb{K}$ is a nonempty set X (later often denotes by V, U or also W). The space X contains elements $x_1, x_2, \ldots$ which are the so-called **vectors**. We define two algebraic operations:*

- *Vector addition: $x + y$ for $x, y \in X$.*
- *Multiplication of vectors with scalars: αx for $x \in X$ and $\alpha \in \mathbb{K}$.*

These operations satisfy the usual laws that they are commutative, associative, and satisfy distributive laws.

3.10 Scalar product

Let V be a vector space over $\mathbb{R}$ (note $\mathbb{C}$ would work as well). A mapping $(\cdot, \cdot) : V \times V \to \mathbb{R}$ is a scalar product (or inner product) if

1. $(u + v, w) = (u, w) + (v, w)$ for all $u, v, w \in V$
2. $(\alpha u, v) = \alpha(u, v)$ for all $u, v \in V$ and $\alpha \in \mathbb{R}$
3. $(u, v) = \overline{(v, u)}$ for all $u, v \in V$
4. $(u, u) \geq 0$ for all $u \in V$
5. $(u, u) = 0$ if and only if when $u = 0$.

Moreover:

1. $(u, v + w) = (u, v) + (u, w)$ for all $u, v, w \in V$
2. $(u, \alpha v) = \bar{\alpha}(u, v)$ for all $u, v \in V$ and $\alpha \in \mathbb{R}$

For real-valued functions, the scalar product is **bilinear** and in the complex-valued case the scalar product is called **sesquilinear**.

It holds:

Proposition 3.9 (Cauchy-Schwarz)**.** *Let V be a vector space with $(\cdot, \cdot)$. Then, it holds*

$$|(u, v)| \leq \|u\|\|v\| \quad \text{for all } u, v \in V.$$

3.11 Normed spaces

Let X be a linear space. The mapping $||\cdot|| : X \to \mathbb{R}$ is a **norm** if

i)	$\|\|x\|\| \geq 0 \quad \forall x \in X$	(Positivity)
ii)	$\|\|x\|\| = 0 \Leftrightarrow x = 0$	(Definiteness)
iii)	$\|\|\alpha x\|\| = \|\alpha\| \, \|\|x\|\|, \quad \alpha \in \mathbb{K}$	(Homogenity)
iv)	$\|\|x + y\|\| \leq \|\|x\|\| + \|\|y\|\|$	(Triangle inequality)

A space X is a normed space when the norm properties are satisfied. If condition ii) is not satisfied, the mapping is called a **semi-norm** and denoted by $|x|_X$ for $x \in X$.

Definition 3.10. *Let $\|\cdot\|$ be a norm on X. Then $\{X, \|\cdot\|\}$ is called a (real) normed space.*

Example 3.11. *We provide some examples:*

1. $\mathbb{R}^n$ *with the euclidian norm* $\|x\| = (\sum_{i=1}^n x_i^2)^{1/2}$ *is a normed space.*

2. *Let* $\Omega := [a, b]$. *The space of continuous functions* $C(\Omega)$ *endowed with the* ***maximum norm***

$$\|u\|_{C(\Omega)} = \max_{x\in\Omega} \|u(x)\|$$

is a normed space.

3. *The space* $\{C(\Omega), \|\cdot\|_{L^2}\}$ *with*

$$\|u\|_{L^2} = \Big(\int_\Omega u(x)^2\, dx\Big)^{1/2}$$

is a normed space.

Definition 3.12. *Two norms are equivalent if converging sequences have the same limits.*

Proposition 3.13. *Two norms* $||\cdot||_A, ||\cdot||_B$ *on* X *are equivalent if and only if there exist two constants* $C_1, C_2 > 0$ *such that*

$$C_1||x||_A \le ||x||_B \le C_2||x||_A \quad \forall x \in X$$

The limits are the same.

Proof. See e.g., [134]. □

Remark 3.14. *These statements have indeed some immediate consequences. For instance, often convergence of an iterative scheme is proven with the help of the Banach fixed point scheme in which the contraction constant* q *must be smaller than* 1*; see for instance Section 10. It is important that not all norms may satisfy* $q < 1$*, but when different norms are equivalent and we pick one that satisfy* $q < 1$*, we can prove convergence.*

3.12 Linear mappings

Let $\{U, \|\cdot\|_U\}$ and $\{V, \|\cdot\|_V\}$ be normed spaces over $\mathbb{R}$.

Definition 3.15 (Linear mappings)**.** *A mapping* $T : U \to V$ *is called* ***linear*** *or* ***linear operator*** *when*

$$T(u) = T(au_1 + bu_2) = aT(u_1) + bT(u_2),$$

for $u = au_1 + bu_2$ *and for* $a, b \in \mathbb{R}$.

Example 3.16. *We discuss two examples:*

1. *Let* $T(u) = \Delta u$. *Then:*

$$T(au_1 + bu_2) = \Delta(au_1 + bu_2) = a\Delta u_1 + b\Delta u_2 = aT(u_1) + bT(u_2).$$

Thus, T *is linear.*

2. *Let* $T(u) = (u \cdot \nabla)u$. *Then:*

$$T(au_1 + bu_2) = ((au_1 + bu_2) \cdot \nabla)(au_1 + bu_2) \neq a(u_1 \cdot \nabla)u_1 + b(u_2 \cdot \nabla)u_2 = aT(u_1) + bT(u_2).$$

Here, T *is nonlinear.*

Exercise 1. *Classify* $T(u) = |u|$.

Definition 3.17 (Linear functional)**.** *A mapping* $T : U \to V$ *with* $V = \mathbb{R}$ *is called* ***linear functional****.*

Definition 3.18. *A mapping* $T : U \to V$ *is called* ***continuous*** *when*

$$\lim_{n\to\infty} u_n = u \quad \Rightarrow \quad \lim_{n\to\infty} Tu_n = Tu.$$

Definition 3.19. *A linear operator $T : U \to V$ is called **bounded**, when the following estimate holds true:*

$$\|Tu\|_V \leq c\|u\|_U.$$

Theorem 3.20. *A linear operator is bounded if and only if it is continuous.*

Proof. See [134], Satz II.1.2. □

Definition 3.21. *Let $T : U \to V$ be a linear and continuous operator. The norm of T is defined as*

$$\|T\| = \sup_{\|u\|_U=1} \|Tu\|_V.$$

Since a linear T is bounded (when continuous) there exists $\|Tu\|_V \leq c\|u\|_U$ for all $u \in U$. The smallest number for c is $c = \|T\|$.

Definition 3.22. *The linear space of all linear and bounded operators from U to V is denoted by*

$$L(U, V).$$

The norm of $L(U, V)$ is the operator norm $\|T\|$.

Definition 3.23 (Dual space)**.** *The linear space of all linear, bounded functionals on U (see Def. 3.17) is the dual space, denoted by U^*, i.e.,*

$$U^* = L(U, \mathbb{R}).$$

For $f \in U^*$, the norm is given by:

$$\|f\|_{U^*} = \sup_{\|u\|_U=1} |f(u)|.$$

The dual space is always a Banach space; see again [134]. For more details on Banach spaces, we also refer to Section 7.1 of these lecture notes.

3.13 Little o and big O - the Landau symbols

Definition 3.24 (Landau symbols)**.** (i) *Let $g(n)$ a function with $g \to \infty$ for $n \to \infty$. Then $f \in O(g)$ if and only if when*

$$\limsup_{n\to\infty} \left|\frac{f(n)}{g(n)}\right| < \infty$$

and $f \in o(g)$ if and only if

$$\lim_{n\to\infty} \left|\frac{f(n)}{g(n)}\right| = 0.$$

(ii) *Let $g(h)$ a function with $g(h) \to 0$ for $h \to 0$. As before, we define $f \in O(g)$ and $f \in o(g)$:*

$$\limsup_{h\to 0} \left|\frac{f(h)}{g(h)}\right| < \infty \quad \Leftrightarrow \quad f \in O(g),$$

and

$$\lim_{h\to 0} \left|\frac{f(h)}{g(h)}\right| = 0 \quad \Leftrightarrow \quad f \in o(g).$$

(iii) *Specifically:*

$$\limsup_{h\to 0} \big|f(h)\big| < \infty \quad \Leftrightarrow \quad f \in O(1),$$

and

$$\lim_{h\to 0} \big|f(h)\big| = 0 \quad \Leftrightarrow \quad f \in o(1).$$

Often, the notation $f = O(g)$ is used rather than $f \in O(g)$ and similarily $f = o(g)$ rather than $f \in o(g)$.

Example 3.25. *Seven examples:*

1. $\frac{1}{x} = \mathrm{o}(\frac{1}{x^2}) \quad (x \to 0)$,
2. $\frac{1}{x^2} = \mathrm{o}(\frac{1}{x}) \quad (|x| \to \infty)$,
3. $e^{-x} = \mathrm{o}(x^{-26}) \quad (x \to \infty)$.
4. *Let* $\varepsilon \to 0$ *and* $h \to 0$. *We write*
$$h = o(\varepsilon)$$
when
$$\frac{h}{\varepsilon} \to 0 \quad \textit{for } h \to 0, \quad \varepsilon \to 0,$$
which means that h *tends faster to* 0 *than* ε.
5. *Let us assume that we have the error estimate (see sections on the numerical analysis)*
$$\|y(t_n) - y_n\|_2 = O(k).$$
Here the O *notation means nothing else than*
$$\frac{\|y(t_n) - y_n\|_2}{k} \to C \quad \textit{for } k \to 0.$$
6. *Let*
$$c_1 k^1 + c_2 k^2 + c_3 k^3 + \ldots$$
be a development in powers of k. *We can shortly write:*
$$c_1 k^1 + O(k^2).$$
7. *Complexity of algorithms, e.g., LR*
$$O(n^3) \quad \textit{for} \quad n \to \infty.$$

Here the fraction converges to a constant C *(and not necessarily* 0*!), which illustrates that* $O(\cdot)$ *convergence is weaker than* $o(\cdot)$ *convergence. On the other hand, this should not yield the wrong conclusion that* $\|y(t_n) - y_n\|_2$ *may not tend to zero. Since* $k \to 0$, *also* $\|y(t_n) - y_n\|_2 \to 0$ *must hold necessarily.*

3.14 Taylor expansion

Let $f : \mathbb{R} \to \mathbb{R}$ be of class $\mathcal{C}^n$ around some point a. Then, the Taylor-Young formula is defined as

$$Tf(x) = \sum_{j=0}^{n} \frac{f^{(j)}(a)}{j!}(x-a)^j + o(|x-a|^n).$$

The remainder $o(|x-a|^n)$ (Landau notation) is a function such that

$$\lim_{x \to a} \frac{o(|x-a|^n)}{|x-a|^n} = 0.$$

Furthermore, we have the Taylor-Lagrange formula. Let f be of class $\mathcal{C}^{n+1}$:

$$Tf(x) = \sum_{j=0}^{n} \frac{f^{(j)}(a)}{j!}(x-a)^j + \frac{f^{(n+1)}(y)}{(n+1)!}(x-a)^{n+1},$$

for some intermediate point y between a and x.

Finally, we recall the Fourier power series. Let $f : \mathbb{R} \to \mathbb{R}$ be of class $\mathcal{C}^\infty$. The power series

$$Tf(x) = \sum_{j=0}^{\infty} \frac{f^{(j)}(a)}{j!}(x-a)^j$$

is the Taylor series of f in the point a.

3.15 Transformation of integrals: substitution rule / change of variables

One of the most important formulas in continuum mechanics and variational formulations is the substitution rule that allows to transform integrals from one domain to another.

In 1D it holds:

Proposition 3.26 (Substituion rule in 1D)**.** *Let $I = [a, b]$ be given. To transform this interval to a new interval, we use a mapping $T(I) = [\alpha, \beta]$ with $T(a) = \alpha$ and $T(b) = \beta$. If $T \in C^1$ (a continuously differentiable mapping) and monotonically increasing (i.e., $T' > 0$), we have the transformation rule:*

$$\int_\alpha^\beta f(y)\,dy = \int_{T(a)}^{T(b)} f(y)\,dy = \int_a^b f(T(x))\,T'(x)\,dx.$$

Proof. Any analysis book. Here Analysis 2, Rolf Rannacher, Heidelberg University [105]. □

Remark 3.27. *In case that $T' < 0$ the previous proposition still holds true, but with a negative sign:*

$$\int_\alpha^\beta f(y)\,dy = \int_{T(b)}^{T(a)} f(y)\,dy = -\int_{T(a)}^{T(b)} f(y)\,dy = \int_a^b f(T(x))\,(-T'(x))\,dx.$$

For both cases with $T' \neq 0$ the formula works and finally yields:

Theorem 3.28. *Let $I = [a, b]$ be given. To transform this interval to a new interval $[\alpha, \beta]$, we employ a mapping T. If $T \in C^1$ (a continuously differentiable mapping) and $T' \neq 0$, it holds:*

$$\int_{T(I)} f(y)\,dy := \int_\alpha^\beta f(y)\,dy = \int_a^b f(T(x))\,|T'(x)|\,dx =: \int_I f(T(x))\,|T'(x)|\,dx.$$

Proof. Any analysis book. Here Analysis 2, Rolf Rannacher, Heidelberg University [105]. □

Remark 3.29. *We observe the relation between the integration increments:*

$$dy = |T'(x)|\,dx.$$

Example 3.30. *Let T be a affine linear transformation defined as*

$$T(x) = ax + b.$$

Then,

$$dy = |a|dx.$$

In higher dimensions, we have the following result of the substitution rule (also known as **change of variables** under the integral):

Theorem 3.31. *Let $\Omega \subset \mathbb{R}^n$ be an open, measurable, domain. Let the function $T : \Omega \to \mathbb{R}$ be of class C^1, one-to-one (injective) and Lipschitz continuous. Then:*

- *The domain $\hat{\Omega} := T(\Omega)$ is measurable.*
- *The function $f(T(\cdot))|detT'(\cdot)| : \Omega \to \mathbb{R}$ is (Riemann)-integrable.*
- *For all measurable subdomains $M \subset \Omega$ it holds the substitution rule:*

$$\int_{T(M)} f(y)\,dy = \int_M f(T(x))|detT'(x)|\,dx,$$

and in particular as well for $M = \Omega$.

Proof. Any analysis book. See e.g., [105] or [86][Chapter 9]. □

Remark 3.32. *In continuum mechanics, T' is the so-called **deformation gradient** and $J := det(T')$, the **volume ratio**.*

3.16 Gauss-Green theorem / Divergence theorem

The Gauss-Green theorem or often known as **divergence theorem**, is one of the most useful formulas in continuum mechanics and numerical analysis.

Let $\Omega \subset \mathbb{R}^n$ an bounded, open domain and $\partial\Omega$ of class C^1.

Theorem 3.33 (Gauss-Green theorem / Divergence theorem)**.** *Suppose that $u := u(x) \in C^1(\bar{\Omega})$ with $x = (x_1, \ldots, x_n)$. Then:*

$$\int_\Omega u_{x_i}\, dx = \int_{\partial\Omega} u n_i\, ds, \quad \text{for } i = 1, \ldots, n.$$

In compact notation, we have

$$\int_\Omega div\, u\, dx = \int_{\partial\Omega} u \cdot n\, ds$$

for each vector field $u \in C^1(\bar{\Omega}; \mathbb{R}^n)$.

Proof. The proof is nontrivial. See for example [86]. □

3.17 Integration by parts and Green's formulae

One of the most important formulae in applied mathematics, physics and continuum mechanics is **integration by parts**.

From the divergence Theorem 3.33, we obtain immediately:

Proposition 3.34 (Integration by parts)**.** *Let $u, v \in C^1(\bar{\Omega})$. Then:*

$$\int_\Omega u_{x_i} v\, dx = -\int_\Omega u v_{x_i}\, dx + \int_{\partial\Omega} uvn_i\, ds, \quad \text{for } i = 1, \ldots, n.$$

In compact notation:

$$\int_\Omega \nabla u v\, dx = -\int_\Omega u \nabla v\, dx + \int_{\partial\Omega} uvn\, ds.$$

Proof. Apply the divergence theorem to uv. Exercise. □

We obtain now some further results, which are very useful, but all are based directly on the integration by parts. For this reason, it is more important to know the divergence theorem and integration by parts formula.

Proposition 3.35 (Green's formulas)**.** *Let $u, v \in C^2(\bar{\Omega})$. Then:*

$$\int_\Omega \Delta u\, dx = \int_{\partial\Omega} \partial_n u\, ds,$$
$$\int_\Omega \nabla u \cdot \nabla v\, dx = -\int_\Omega \Delta u\, v\, dx + \int_{\partial\Omega} v\, \partial_n u\, ds.$$

Proof. Apply integration parts. □

Proposition 3.36 (Green's formula in 1D)**.** *Let $\Omega = (a, b)$. Let $u, v \in C^2(\bar{\Omega})$. Then:*

$$\int_\Omega u'(x) \cdot v'(x)\, dx = -\int_\Omega u''(x)\, v(x)\, dx + [u'(x)v(x)]_{x=a}^{x=b}$$

Proof. Apply integration parts. □

3.18 Main notation

Notation of scalar products, linear forms and bilinear forms. Let Ω be a domain in $\mathbb{R}^n$ with the usual properties to ensure integrability and differentiability:

1. Let $x, y \in V \subset R^n$. Then the scalar product is denoted by:

$$(x, y) = \int_\Omega xy \, dz \quad x, y \in V.$$

2. Let $X, Y \in V \subset R^{n \times n}$. Then the scalar product for a matrix is denoted by:

$$(X, Y) = \int_\Omega X : Y \, dz \quad X, Y \in V.$$

   Here, : stands for the Frobenius scalar product.

3. Often, we use: Find $u \in V$ such that

$$a(u, \varphi) = l(\phi) \quad \forall \varphi \in V.$$

   Here $a(u, \varphi) : V \times V \to \mathbb{R}$ is a bilinear form and $l(\varphi) \in V^*$ is a linear form (linear functional).

4. For nonlinear problems, the solution variable $u \in V$ is nonlinear while the test function is still linear. Here we use the notation: Find $u \in V$ such that

$$a(u)(\varphi) = l(\varphi) \quad \forall \varphi \in V.$$

   Here, $a(u)(\varphi)$ is a so-called semi-linear form.

5. For (linear) PDE systems, my notation is: Find $U \in X$ such that

$$A(U, \Psi) = F(\Psi) \quad \forall \Psi \in X.$$

6. For nonlinear PDE systems, my notation is: Find $U \in X$ such that

$$A(U)(\Psi) = F(\Psi) \quad \forall \Psi \in X.$$

3.19 Chapter summary and outlook

Having the notation setup, we shall now begin with the specific topics promised in the title of these lecture notes. To this end, we are going to have an extended introduction in which mathematical modeling of some differential equations is carried out.

4 An extended introduction

In this introduction (despite being already in Chapter 4), we first start with modeling on how the Laplace equation can be derived from physical fundamental principles.

4.1 Examples of differential equations

We start with some examples:

4.1.1 Variational principles in mechanics

Variational principles were developed in physics and more precisely in **classical mechanics**, e.g., [52, 61].

One of the first variational problems was designed by Jacob Bernoulli in the year 1696: how does a mass point reach from a point p_1 in the shortest time T another point p_2 under graviational forces? In consequence, we seek a minimal time T along a curve $u(x)$:

$$\min J(u) = \min(T)$$

with respect to the boundary conditions $u(x_1) = u_1$ and $u(x_2) = u_2$. To obtain the solution $u(x)$, we start from energy conservation, which yields for the kinetic and the potential energies:

$$\frac{mv^2}{2} = mg(u_1 - u),$$

where m is the mass, v the velocity of the mass, g the graviational force, u_1 the boundary condition, and $u = u(x)$ the sought solution curve. We have the functional:

$$J(u) = T = \int_1^2 \frac{ds}{v} = \int_{x_1}^{x_2} \sqrt{\frac{1 + u'(x)^2}{2g(u_1 - u(x))}}\, dx.$$

Proposition 4.1. *The solution to this problem is the so-called **Brachistochrone** formulated by Jacob Bernoulli in 1696 [52].*

More generally, we formulate the unconstrained problem:

Formulation 4.2. *Find $u = u(x)$ such that*

$$\min J(u)$$

with

$$J(u) = \int_{x_1}^{x_2} F(u, u', x)\, dx.$$

Here, we assume that the function $F(u, u', x)$ and the boundary values $u(x_1) - u_1$ and $u(x_2) = u_2$ are known.

The idea is that we vary $J(u + \delta u)$ with a small increment δu as we have formulated in Section 8.1.2. Then, we arrive at the **Euler-Lagrange equations**:

Definition 4.3. *The (strong form of the) Euler-Lagrange equations are obtained as stationary point of the functional $J(u)$ and are nothing else, but the PDE in differential form:*

$$\frac{d}{dx}\frac{\partial F(u, u', x)}{\partial u'} = \frac{\partial F(u, u', x)}{\partial u}.$$

Remark 4.4 (Weak form of Euler-Lagrange equations)**.** *An equivalent statement is related to the weak form of the Euler-Lagrange equations in Banach spaces. Here, the functional $J(u)$ is differentiated w.r.t. u into the direction ϕ yielding $J'_u(u)(\phi)$ as used in Section 8.1.2.*

Example 4.5. *To compute the length of a curve $u(x)$ between two points $(x_1, u(x_1))$ and $(x_2, u(x_2))$, the following functional is used:*

$$J(u) = \int_1^2 ds = \int_{x_1}^{x_2} \sqrt{1 + (u')^2}\, dx.$$

This brings us to the question for which function $u(x)$ the functional $J(u)$ attains a minimum, i.e., measuring the shortest distance between $(x_1, u(x_1))$ and $(x_2, u(x_2))$. This functional is the starting point to derive the clothesline problem.

Moreover, we identify $F(u, u', x) = \sqrt{1 + (u')^2}$. The Euler-Lagrange equation is then given by

$$\frac{d}{dx} \frac{u'(x)}{\sqrt{1 + (u')^2}} = 0.$$

When no external forces act (right hand side is zero), we can immediately derive the solution:

$$\frac{d}{dx} \frac{u'(x)}{\sqrt{1 + (u')^2}} = 0 \quad \Rightarrow \quad \frac{d}{dx} u'(x) = 0 \quad \Rightarrow \quad u'(x) = const \quad \Rightarrow \quad u(x) = ax + c.$$

The boundary conditions (not specified here) will determine the constants a and c. Of course, the solution, i.e., the shortest distance between two points, is a linear function. When external forces act (e.g., gravity), we will arrive at a solution similar to that in Section 8.1.1.

4.1.2 Population growth

Let us compute the growth of a species (for example human beings) with a very simple (and finally not that realistic) model. But this shows that reality can be represented to some extend at least by simple models, but that continuous comparisons with other data is also necessary. Furthermore, this also shows that mathematical modeling often starts with a simple equation and is then continuously further augmented with further terms and coefficients. In the final end we arrive at complicated formulae.

To get started, let us assume that the population number is $y = y(t)$ at time t. Furthermore, we assume constant growth g and mortalities rates m, respectively. In a short time frame dt we have a relative increase of

$$\frac{dy}{y} = (g - m)dt$$

of the population. Re-arranging and taking the limit $dt \to 0$ yields:

$$\lim_{t \to 0} \frac{dy}{dt} = (g - m)y$$

and thus

$$y' = (g - m)y.$$

This is the so-called **Malthusian law of growth**. For this ordinary differential equation (ODE), we can even explicitely compute the solution:

$$y(t) = c \exp((g - m)(t - t_0)).$$

This can be achieved with separation of variables:

$$y' = \frac{dy}{dt} = (g - m)y \tag{2}$$

$$\Rightarrow \int \frac{dy}{y} = \int_{t_0}^{t} (g - m)dt \tag{3}$$

$$\Rightarrow Ln|y| + C = (g - m)(t - t_0) \tag{4}$$

$$\Rightarrow y = \exp[C] \cdot \exp[(g - m)(t - t_0)] \tag{5}$$

In order to work with this ODE and to compute the future development of the species we need an initial value at some starting point t_0:

$$y(t_0) = y_0.$$

With this value, we can further work to determine the constant $\exp(C)$:

$$y(t_0) = \exp(C)\exp[(g-m)(t_0-t_0)] = \exp(C) = y_0. \tag{6}$$

Let us say in the year $t_0 = 2011$ there have been two members of this species: $y(2011) = 2$. Supposing a growth rate of 25 per cent per year yields $g = 0.25$. Let us say $m = 0$ - nobody will die. In the following we compute two estimates of the future evolution of this species: for the year $t = 2014$ and $t = 2022$. We first obtain:

$$y(2014) = 2\exp(0.25 * (2014 - 2011)) = 4.117 \approx 4.$$

Thus, four members of this species exist after three years. Secondly, we want to give a 'long term' estimate for the year $t = 2022$ and calculate:

$$y(2022) = 2\exp(0.25 * (2022 - 2011)) = 31.285 \approx 31.$$

In fact, this species has an increase of 29 members within 11 years. If you translate this to human beings, we observe that the formula works quite well for a short time range but becomes somewhat unrealistic for long-term estimates though.

Remark 4.6. *An improvement of the basic population model is the* ***logistic differential equation*** *that was proposed by Verhulst; see e.g., [25]. In equations, this reads:*

$$y' = ay - by^2, \quad y(t_0) = y_0.$$

which is obviously a nonlinear equation. For population growth, usually $a \gg b$. For instance, one example is for the growth of the world population is [25] (as of the year 1993):

$$t_0 = 1965, \quad y_0 = 3.34 \times 10^9, \quad a = 0.029, \quad b = 2.695 \times 10^{-12}.$$

4.1.3 Laplace equation / Poisson problem

This equation has several applications, e.g., Fick's law of diffusion or heat conduction (temporal diffusion) or the deflection of a solid membrane:

Formulation 4.7 (Laplace problem / Poisson problem)**.** *Let Ω be an open set. The* ***Laplace equation*** *reads:*

$$-\Delta u = 0 \quad \text{in } \Omega.$$

The ***Poisson problem*** *reads:*

$$-\Delta u = f \quad \text{in } \Omega.$$

Definition 4.8. *A C^2 function (C^2 means two times continuously differentiable) that satisfies the Laplace equation is called* ***harmonic*** *function.*

The physical interpretation is as follows. Let u denote the density of some quantity, for instance concentration or temperature, in equilibrium. If G is any smooth region $G \subset \Omega$, the flux F of the quantity u through the boundary ∂G is zero:

$$\int_{\partial G} F \cdot n \, dx = 0. \tag{7}$$

Here F denotes the flux density and n the outer normal vector. Gauss' divergence theorem yields:

Formulation 4.9 (Integral conservation equation)**.** *Extending* (7) *to Ω we have*

$$\int_{\partial\Omega} F \cdot n \, dx = 0.$$

Employing Gauss' divergence theorem, it holds

$$\int_{\Omega} \nabla \cdot F \, dx = 0. \tag{8}$$

This integral form of a conservation equation is very often the initial representation of physical laws.

The question is now how we come from the integral form to a pointwise differential equation. We first need to assume that the integrand is sufficiently regular. Here, F should be continuously differentiable. Second, we request that (8) holds for arbitrary $G \subset \Omega$ as previously mentioned. Then, it can be inferred from results in analysis (calculus) [85, 86] that the integrand is pointwise zero:

Formulation 4.10 (Conservation equation in differential form)**.**

$$\nabla \cdot F = 0 \quad in\ \Omega. \tag{9}$$

Constitutive laws Now we need a second assumption (or better a relation) between the flux and the quantity u. Such relations do often come from material properties and are so-called **constitutive laws.** In many situations it is reasonable to assume that the flux F is proportional to the negative gradient $-\nabla u$ of the quantity u. This means that flow goes from regions with a higher concentration to lower concentration regions. For instance, the rate at which energy 'flows' (or diffuses) as heat from a warm body to a colder body is a function of the temperature difference. The larger the temperature difference, the larger the diffusion. We consequently obtain as further relation:

$$F = -\nabla u.$$

Plugging into the Equation (9) yields:

$$\nabla \cdot F = \nabla \cdot (-\nabla u) = -\nabla \cdot (\nabla u) = -\Delta u = 0.$$

This is the simplest derivation one can make. Adding more knowledge on the underlying material of the body, a material parameter $a > 0$ can be added:

$$\nabla \cdot F = \nabla \cdot (-a\nabla u) = -\nabla \cdot (a\nabla u) = -a\Delta u = 0.$$

And adding a nonconstant and spatially dependent material further yields:

$$\nabla \cdot F = \nabla \cdot (-a(x)\nabla u) = -\nabla \cdot (a(x)\nabla u) = 0.$$

In this last equation, we do not obtain any more the classical Laplace equation but a diffusion equation in divergence form.

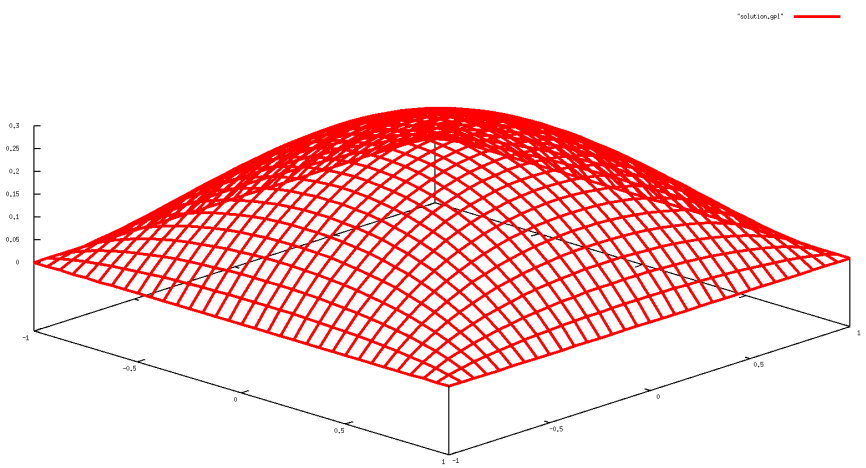


Figure 3: Solution of the Poisson problem on $\Omega = (-1, 1)^2$ with the right hand side $f = 1$.

4.1.4 Mass conservation / First-order hyperbolic problem

In this next example let us again consider some concentration, but now a time dependent situation, which brings us to a first-order hyperbolic equation. The application might be transport of a species/concentration in some fluid flow (e.g. water) or nutrient transport in blood flow. Let $\Omega \subset \mathbb{R}^n$ be an open domain, $x \in \Omega$ the spatial variable and t the time. Let $\rho(x,t)$ be the density of some quantity (e.g., concentration) and let $v(x,t)$ its velocity. Then the vector field

$$F = \rho v \quad \text{in } \mathbb{R}^n$$

denotes the flux of this quantity. Let G be a subset of Ω. Then we have as in the previous case (Equation (7)) the definition:

$$\int_{\partial G} F \cdot n \, dx.$$

But this time we do not assume the 'equilibrium state' but 'flow'. That is to say that the outward flow through the boundary ∂G must coincide with the temporal decrease of the quantity:

$$\int_{\partial G} F \cdot n \, ds = -\frac{d}{dt} \int_G \rho \, dx.$$

We then apply again Gauss' divergence theorem to the left hand side and bring the resulting term to the right hand side. We now encounter a principle difficulty that the domain G may depend on time and consequently integration and differentiation do not commute. Therefore, in a first step we need to transform the integrand of

$$\frac{d}{dt} \int_G \rho \, dx$$

onto a fixed reference configuration $\hat{G}$ in which we can insert the time derivative under the integral sign. Then we perform the calculation and transform lastly everything back to the physical domain G. Let the mapping between $\hat{G}$ and G be denoted by T. Then, it holds:

$$x \in G : x = T(\hat{x}, t), \quad \hat{x} \in \hat{G}.$$

Moreover, $dx = J(\hat{x},t)d\hat{x}$, where $J := \det(\nabla T)$. Using the substitution rule (change of variables) in higher dimensions (see Section 3.15) yields:

$$\frac{d}{dt} \int_G \rho(x,t) \, dx = \frac{d}{dt} \int_{\hat{G}} \rho(T(\hat{x},t),t) J(\hat{x},t) d\hat{x}.$$

We eliminated time dependence on the right hand side integral and thus differentiation and integration commute now:

$$\int_{\hat{G}} \frac{d}{dt} \Big(\rho(T(\hat{x},t),t) J(\hat{x},t) \Big) d\hat{x} = \int_{\hat{G}} \Big(\frac{d}{dt} \rho(T(\hat{x},t),t) \cdot J(\hat{x},t) + \rho(T(\hat{x},t),t) \frac{d}{dt} J(\hat{x},t) \Big) d\hat{x}$$

Here, $\frac{d}{dt}\rho(T(\hat{x},t),t)$ is the material time derivative of a spatial field (see e.g., [76]), which is <u>not</u> the same as the partial time derivative! In the last step, we need the Eulerian expansion formula

$$\frac{d}{dt} J = \nabla \cdot v \, J.$$

Then:

$$\begin{aligned}
&\int_{\hat{G}} \Big(\frac{d}{dt}\rho(T(\hat{x},t),t)\cdot J(\hat{x},t) + \rho(T(\hat{x},t),t)\frac{d}{dt}J(\hat{x},t)\Big) d\hat{x} \\
&= \int_{\hat{G}} \Big(\frac{d}{dt}\rho(T(\hat{x},t),t)\cdot J(\hat{x},t) + \rho(T(\hat{x},t),t)\nabla\cdot v\, J\Big) d\hat{x} \\
&= \int_{\hat{G}} \Big(\frac{d}{dt}\rho(T(\hat{x},t),t) + \rho(T(\hat{x},t),t)\nabla\cdot v\Big) J(\hat{x},t) d\hat{x} \\
&= \int_{G} \Big(\frac{d}{dt}\rho(x,t) + \rho(x,t)\nabla\cdot v\Big) dx \\
&= \int_{G} \Big(\partial_t\rho(x,t) + \nabla\rho(x,t)\, v + \rho(x,t)\nabla\cdot v\Big) dx \\
&= \int_{G} \Big(\partial_t\rho(x,t) + \nabla\cdot(\rho v)\Big) dx.
\end{aligned}$$

Collecting the previous calculations brings us to:

$$\int_G (\partial_t\rho + \nabla\cdot F)\, dx = \int_G (\partial_t\rho + \nabla\cdot(\rho v)\, dx, \quad \text{where } F = \rho v \text{ as introduced before.}$$

This is the so-called continuity equation (or mass conservation). Since G was arbitrary, we are allowed to write the strong form:

$$\partial_t\rho + \nabla\cdot(\rho v) = 0 \quad \text{in } \Omega. \tag{10}$$

If there are sources or sinks, denoted by f, inside Ω we obtain the more general formulation:

$$\partial_t\rho + \nabla\cdot(\rho v) = f \quad \text{in } \Omega.$$

On the other hand, various simplifications of Equation (10) can be made when certain requirements are fulfilled. For instance, if ρ is not spatially varying, we obtain:

$$\partial_t\rho + \rho\nabla\cdot v = 0 \quad \text{in } \Omega.$$

If furthermore ρ is constant in time, we obtain:

$$\nabla\cdot v = 0 \quad \text{in } \Omega.$$

This is now the mass conservation law that appears for instance in the incompressible Navier-Stokes equations, which are discussed a little bit later below.

In terms of the density ρ, we have shown in this section:

Theorem 4.11 (Reynolds' transport theorem)**.** *Let $\Phi := \Phi(x,t)$ be a smooth scalar field and Ω a (moving) domain. It holds:*

$$\frac{d}{dt}\int_\Omega \Phi\, dx = \int_{\partial\Omega} \Phi v\cdot n\, ds + \int_\Omega \frac{\partial\Phi}{\partial t}\, dx$$

*The first term on the right hand side represents the **rate of transport** (also known as the **outward normal flux**) of the quantity Φv across the boundary surface $\partial\Omega$. This contribution originates from the moving domain Ω. The second contribution is the **local time rate of change** of the spatial scalar field Φ. If the domain Ω does not move, the first term on the right hand side will of course vanish. Using Gauss' divergence theorem it holds furthermore:*

$$\frac{d}{dt}\int_\Omega \Phi\, dx = \int_\Omega \nabla\cdot(\Phi\, v)\, dx + \int_\Omega \frac{\partial\Phi}{\partial t}\, dx.$$

4.1.5 Elasticity (Lamé-Navier)

This example is already difficult because a system of nonlinear equations is considered:

Formulation 4.12. *Let $\widehat{\Omega}_s \subset \mathbb{R}^n, n = 3$ with the boundary $\partial\widehat{\Omega} := \widehat{\Gamma}_D \cup \widehat{\Gamma}_N$. Furthermore, let $I := (0, T]$ where $T > 0$ is the end time value. The equations for geometrically non-linear elastodynamics in the reference configuration $\widehat{\Omega}$ are given as follows: Find vector-valued displacements $\hat{u}_s := (\hat{u}_s^{(x)}, \hat{u}_s^{(y)}, \hat{u}_s^{(z)}) : \widehat{\Omega}_s \times I \to \mathbb{R}^n$ such that*

$$\begin{aligned}
\hat{\rho}_s \partial_t^2 \hat{u}_s - \widehat{\nabla} \cdot (\widehat{F}\widehat{\Sigma}) &= 0 \quad && \text{in } \widehat{\Omega}_s \times I, \\
\hat{u}_s &= 0 && \text{on } \widehat{\Gamma}_D \times I, \\
\widehat{F}\widehat{\Sigma} \cdot \hat{n}_s &= \hat{h}_s && \text{on } \widehat{\Gamma}_N \times I, \\
\hat{u}_s(0) &= \hat{u}_0 && \text{in } \widehat{\Omega}_s \times \{0\}, \\
\hat{v}_s(0) &= \hat{v}_0 && \text{in } \widehat{\Omega}_s \times \{0\}.
\end{aligned}$$

We deal with two types of boundary conditions: Dirichlet and Neumann conditions. Furthermore, two initial conditions on the displacements and the velocity are required. The constitutive law is given by the geometrically nonlinear tensors (see e.g., Ciarlet [34]):

$$\widehat{\Sigma} = \widehat{\Sigma}_s(\hat{u}_s) = 2\mu\widehat{E} + \lambda tr(\widehat{E})I, \quad \widehat{E} = \frac{1}{2}(\widehat{F}^T\widehat{F} - I). \tag{11}$$

Here, μ and λ are the Lamé coefficients for the solid. The solid density is denoted by $\hat{\rho}_s$ and the solid deformation gradient is $\widehat{F} = \hat{I} + \widehat{\nabla}\hat{u}_s$ where $\hat{I} \in \mathbb{R}^{3\times3}$ is the identity matrix. Furthermore, $\hat{n}_s$ denotes the normal vector.

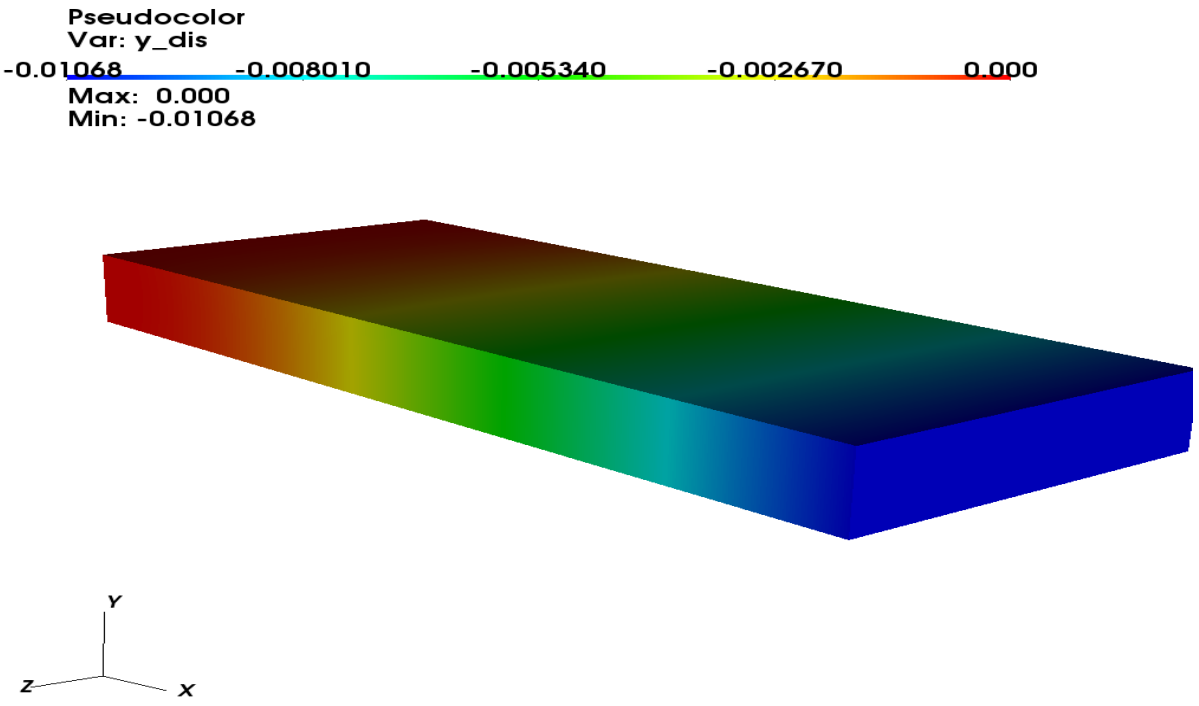


Figure 4: Deformed elastic flag.

4.1.6 The incompressible, isothermal Navier-Stokes equations (fluid mechanics)

Flow equations in general are extremely important and have an incredible amount of possible applications such as for example water (fluids), blood flow, wind, weather forecast, aerodynamics:

Formulation 4.13. *Let $\Omega_f \subset \mathbb{R}^n, n = 3$. Furthermore, let the boundary be split into $\partial\Omega_f := \Gamma_{in} \cup \Gamma_{out} \cup \Gamma_D \cup \Gamma_i$. The isothermal, incompressible (non-linear) Navier-Stokes equations read: Find vector-valued velocities $v_f : \Omega_f \times I \to \mathbb{R}^n$ and a scalar-valued pressure $p_f : \Omega_f \times I \to \mathbb{R}$ such that*

$$\begin{aligned} \rho_f \partial_t v_f + \rho_f v_f \cdot \nabla v_f - \nabla \cdot \sigma_f(v_f, p_f) &= 0 \quad && \text{in } \Omega_f \times I, \\ \nabla \cdot v_f &= 0 && \text{in } \Omega_f \times I, \\ v_f^D &= v_{in} && \text{on } \Gamma_{in} \times I, \\ v_f &= 0 && \text{on } \Gamma_D \times I, \\ -p_f n_f + \rho_f \nu_f \nabla v_f \cdot n_f &= 0 && \text{on } \Gamma_{out} \times I, \\ v_f &= h_f && \text{on } \Gamma_i \times I, \\ v_f(0) &= v_0 && \text{in } \Omega_f \times \{t = 0\}, \end{aligned} \tag{12}$$

where the (symmetric) Cauchy stress is given by

$$\sigma_f(v_f, p_f) := -p_f I + \rho_f \nu_f (\nabla v_f + \nabla v_f^T),$$

with the density ρ_f and the kinematic viscosity ν_f. The normal vector is denoted by n_f.

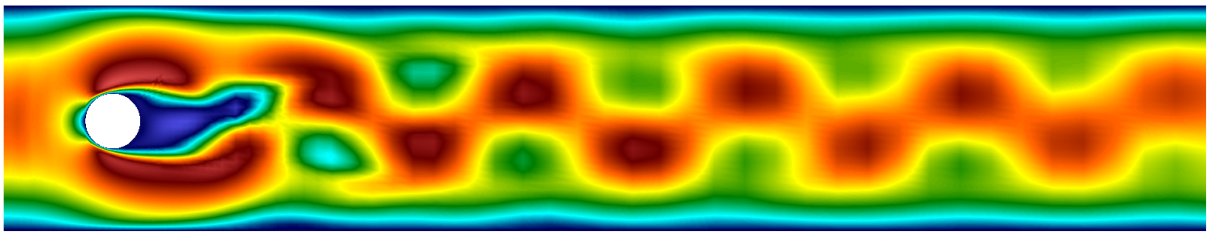


Figure 5: Prototype example of a fluid mechanics problem (isothermal, incompressible Navier-Stokes equations): the famous Karman vortex street. The setting is based on the benchmark setting [118] and the code can be found in NonStat Example 1 in [62] www.dopelib.net.

Remark 4.14. *The two Formulations 4.12 and 4.13 are very important in many applications and their coupling results in fluid-structure interaction. Here we notice that fluid flows are usually modeled in Eulerian coordinates and solid deformations in Lagrangian coordinates. In the case of small displacements, the two coordinate systems can be identified, i.e., $\widehat{\Omega} \simeq \Omega$. This is the reason why in many basic books - in particular basics of PDE theory or basics of numerical algorithms - the 'hat' notation (or similar notation to distinguish coordinate systems) is not used.*

Exercise 2. *Recapitulate (in case you have had classes on continuum mechanics) the differences between Lagrangian and Eulerian coordinates.*

Remark 4.15. *In Section 9.24, we study in more detail the stationary Navier-Stokes equations (NSE) in terms of a well-acknowledged benchmark problem.*

4.1.7 Coupled: Biot (porous media flow) and Biot coupled to elasticity

In this section the Biot model is considered, which is a standard model for porous media applications. In fact, the Biot system [20–22] is a multi-scale problem which is identified on the micro-scale as a fluid-structure interaction problem (details on the interface law are found in Mikelić and Wheeler [95]). Through homogenization, the Biot system is derived for the macro-scale level. This system is specifically suited for applications in subsurface modeling for the poroelastic part, the so-called pay-zone. On the other hand, surrounding rock (the non-pay zone) is modeled with the help of linear elasticity [34]. Therefore, the final configuration belongs to a multiphysics problem in non-overlapping domains.

Remark 4.16. *The Biot system allows for modeling of fluids and solids in porous media. In contrast to 'true' fluid-structure interaction, the displacements are assumed to be small and therefore, all equations are modeled in Eulerian coordinates. Please see literature on solid mechanics [76] or fluid-structure interaction [112] in which Lagrangian or Eulerian coordinates are explained.*

4.1.7.1 Modeling We begin by describing the setting for a pure poroelastic setting, the so-called pay-zone. Let Ω_B the domain of interest and $\partial\Omega_B$ its boundary with the partition:

$$\partial\Omega_B = \Gamma_u \cup \Gamma_t = \Gamma_p \cup \Gamma_f,$$

where Γ_u denotes the displacement boundary (Dirichlet), Γ_t the total stress or traction boundary (Neumann), Γ_p the pore pressure boundary (Dirichlet), and Γ_f the fluid flux boundary (Neumann). Concretely, we have for a material with the displacement variable u and its Cauchy stress tensor σ:

$$\begin{aligned} u &= \bar{u} \quad \text{on } \Gamma_u, \\ \sigma n &= \bar{t} \quad \text{on } \Gamma_t, \end{aligned}$$

for given $\bar{u}$ and $\bar{t}$, and the normal vector n. For the pressure with the permeability tensor K and fluid's viscosity η_f, we have the conditions:

$$\begin{aligned} p &= \bar{p} \quad \text{on } \Gamma_p, \\ -\frac{K}{\eta_f}\Big(\nabla p - \rho_f g\Big) \cdot n &= \bar{q} \quad \text{on } \Gamma_f, \end{aligned}$$

for given $\bar{p}$ and $\bar{q}$; and the density ρ_f and the gravity g. For the initial conditions at time $t = 0$, we prescribe

$$\begin{aligned} p(t=0) &= p_0 \quad \text{in } \Omega_B, \\ \sigma(t=0) &= \sigma_0 \quad \text{in } \Omega_B. \end{aligned}$$

The extension of the previous setting with some non-pay zone Ω_S (e.g., rock modeled as an elastic medium) is displayed in Figure 6. In this case (if Ω_B is totally embedded in Ω_S) the boundary conditions on $\partial\Omega_B$ reduce to interface conditions $\partial\Omega_B := \Gamma_i = \Omega_B \cap \Omega_S$.

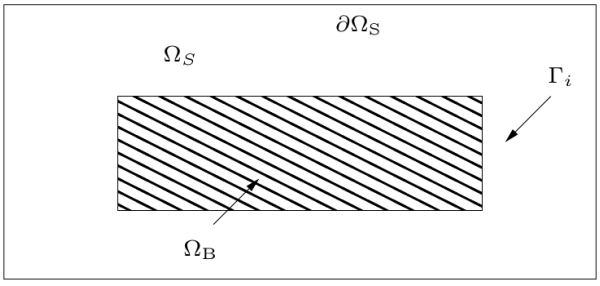


Figure 6: Configuration of the coupled Biot-Lamé-Navier system with the Biot equations in the pay zone Ω_B and linear elasticity in the non-pay zone Ω_S.

Let $I := (0, T)$ denote the time interval. The underlying Biot system reads:

Formulation 4.17. *Find the pressure p_B and displacement u_B such that*

$$\partial_t(c_B p_B + \alpha_B div u_B) - \frac{1}{\eta_f} div\, K(\nabla p_B - \rho_f g) = q \quad in\ \Omega_B \times I,$$
$$-div\big(\sigma_B^{por}(u_B)\big) = f_B \quad in\ \Omega_B \times I,$$

with the total poroelastic stress tensor

$$\sigma_B^{por}(u) := \sigma_B(u_B) - \alpha_B \nabla p_B,$$

with the elastic stress tensor

$$\sigma_B(u_B) := \mu_B e(u_B) + \lambda_B div\, u_B I,$$

and the coefficients $c_B \geq 0$ (related to the compressibility of the fluid; $c_B = 0$ means incompressible fluid), the Biot-Willis constant $\alpha_B \in [0,1]$, (in fact, this constant relates to the amount of coupling between the flow part and the elastic part) and the permeability tensor K, fluid's viscosity and its density η_f and ρ_f, gravity g and a volume source term q (i.e., usually, wells for oil production and fluid injection). In the second equation, the Lamé coefficients are denoted by $\lambda_B > 0$ and $\mu_B > 0$ and f_B is a volume force.

The velocity v_B in the porous medium is obtained with the help of Darcy's law [38] and the Darcy equations which are obtained through homogenization of Stokes's equations [16]. It holds:

$$v_\mathrm{B} = -\frac{1}{\eta_f} K(\nabla p_\mathrm{B} - \rho_f g).$$

Usually the non-pay zone is described in terms of linear elasticity:

Formulation 4.18. *Find a displacement u_S such that*

$$-div\big(\sigma_S(u_S)\big) = f_S \quad in\ \Omega_S \times I,$$

with

$$\sigma_S(u_S) := \mu_S(\nabla u_S + \nabla u_S^T) + \lambda_S div\, u_S I,$$

with the Lamé coefficients μ_S and λ_S and a volume force f_S. On the boundary $\partial\Omega_S := \Gamma_D \cup \Gamma_N$, the conditions

$$u_S = \bar{u}_S \quad on\ \Gamma_D, \quad \sigma_S(u_S) n_S = \bar{t}_S \quad on\ \Gamma_N,$$

are prescribed with given $\bar{u}_S$ and $\bar{t}_S$.

4.1.7.2 Interface conditions It finally remains to describe the interface conditions on Γ_i between the two sub-systems:

$$\begin{aligned} u_\mathrm{B} &= u_S, \\ \sigma_B(u_B) n_B - \sigma_S(u_S) n_S &= \alpha p_\mathrm{B} n_B, \\ -\frac{1}{\eta_f} K(\nabla p_\mathrm{B} - \rho_f g) \cdot n_S &= 0. \end{aligned} \tag{13}$$

It is important to notice that the second condition in (13), requires careful implementation on the interface.

4.1.7.3 The coupled problem in variational form The coupled system of equations is formulated in terms of a variational monolithically-coupled system:

- Domain-coupling: solution variables enter as linear terms into the other problem. Here coupling both equations yields a linear overall problem.
- Interface-coupling on Γ_i between the poroelastic model and the linearized elastic problem.

We first define the function spaces using the boundary conditions from before:

$$V_P := \{p \in H^1(\Omega_B) |\ p = 0 \text{ on } \Gamma_r\},$$
$$V_S := \{u \in H^1(\Omega_B) |\ u_y = 0 \text{ on } \Gamma_b,\ u_x = 0 \text{ on } \Gamma_l\}.$$

The definition of a weak formulation in a variational-monolithic setting reads:

Formulation 4.19 (Biot Problem in Ω_B)**.** *Find $(p, u) \in V_P \times V_S$, with $p(0) = p^0$, such that for almost all times $t \in I$, it holds*

$$\begin{aligned} c_B(\partial_t p, \phi^p) + \alpha_B(\nabla \cdot u, \phi^p) + \frac{K}{\nu_F}(\nabla p, \nabla \phi^p) & \\ -\rho_F(g, \phi^p) - (q, \phi^p) = 0 \quad \forall \phi^p \in V_P, & \\ (\sigma_B, \nabla \phi^u) - \alpha_B(pI, \nabla \phi^u) - \int_{\Gamma_{top}} (\bar{t}_S - \alpha p n)\phi^u \, ds - (f_B, \phi^u) = 0 \quad \forall \phi^u \in V_S. & \end{aligned}$$

By augmenting the domain with an outer elastic part we must account for an additional elastic equation. To this end, we redefine

$$V_S := \{u \in H^1(\Omega_B \cup \Omega_S) |\ u_y = 0 \text{ on } \Gamma_b,\ u_x = 0 \text{ on } \Gamma_{l,B} \cup \Gamma_{l,S}\},$$

in which we now implicitly have built-in the kinematic condition $u_B = u_S$ on Γ_i. The dynamic coupling condition from (13) is implicitly contained in the coupled PDE system.

Formulation 4.20 (Biot-Lamé-Navier Problem in $\Omega_B \cup \Omega_S$)**.** *Find $(p, u) \in V_P \times V_S$, with the initial data $p(0) = p^0$, such that for almost all times $t \in I$, it holds*

$$\begin{aligned} c_B(\partial_t p, \phi^p) + \alpha_B(\nabla \cdot u, \phi^p) + \frac{K}{\nu_F}(\nabla p, \nabla \phi^p) & \\ -\rho_F(g, \phi^p) - (q, \phi^p) = 0 \quad \forall \phi^p \in V_P, & \\ (\sigma_B, \nabla \phi^u) - \alpha_B(pI, \nabla \phi^u) - (f_B, \phi^u) = 0 \quad \forall \phi^u \in V_S(\Omega_B), & \\ (\sigma_S, \nabla \phi^u) - \int_{\Gamma_{top}} \bar{t}_S \phi^u \, ds - (f_S, \phi^u) = 0 \quad \forall \phi^u \in V_S(\Omega_S). & \end{aligned}$$

4.1.7.4 Mandel's problem We now present the results of two test cases. Specifically, we consider the well-known Mandel problem, which is an acknowledged benchmark in subsurface modeling. In the second example, the augmented Mandel problem [56] is used as verification. The configuration and parameters are taken from [55, 66].

Remark 4.21. *The situation has very practical applications: Suppose a sponge filled with water that is squeezed: first the fluid pressure inside increases, but with time, the water will flow out and the pressure will decrease to zero.*

4.1.7.4.1 Configuration The domain is $[0, 100m] \times [0, 20m]$. The initial mesh is 4-times globally refined corresponding to 256 cells and 2467 degrees of freedom.

4.1.7.4.2 Boundary conditions The boundary conditions are given as:

$$\begin{aligned} \sigma_B n - \alpha_B p_B n = \bar{t} - \alpha_B p_B n \quad & \text{on } \Gamma_{top}, \\ p = 0 \quad & \text{on } \Gamma_r, \\ u_y = 0 \quad & \text{on } \Gamma_b, \\ u_x = 0 \quad & \text{on } \Gamma_l. \end{aligned}$$

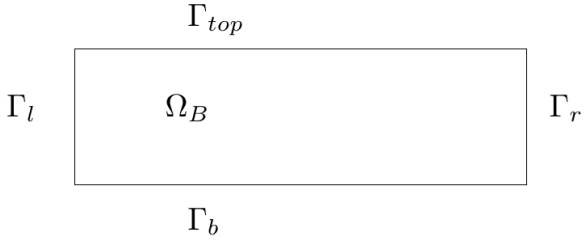


Figure 7: Configuration of Mandel's problem.

4.1.7.4.3 Parameters As parameters, we choose: $M_B = 2.5 * 10^{12} Pa, c_B = 1/M_B Pa^{-1}, \alpha_B = 1.0, \nu_F = 1.0 * 10^{-3} m^2/s, K_B = 100md = 10^{-13} m^2, \rho_F = 1.0 kg/m^3, \bar{t} = F = 1.0 * 10^7 Pa \cdot m$. As elasticity parameters in Biot's model, we use $\mu_S = 1.0e + 8Pa, \nu_S = 0.2$. The time step is chosen as $k = 1000s$. The final time is $5 * 10^6 s$ (corresponds to computing 5000 time steps).

4.1.7.4.4 Discussion of our findings For different time steps $t_1 = 1 * 10^3, t_2 = 5 * 10^3, t_3 = 1 * 10^4, t_4 = 1 * 10^5, t_5 = 5 * 10^5$ and $t_{10} = 5 * 10^6 [s]$ (the numeration of time steps is taken from [55]) the solution of the pressure and the x-displacement evaluated on the x-axis is shown in Figure 8. These values coincides with the literature values displayed in [55, 66, 91, 92]. In particular, the well-known Mandel Creyer effect (non-monotonic pressure behavior over time) is identified.

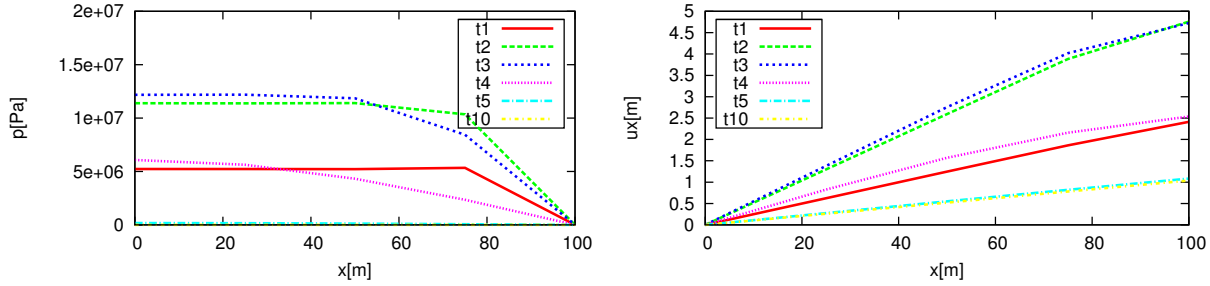


Figure 8: Mandel's problem: Pressure trace and x-displacement on the x-axis.

4.1.7.5 Augmented Mandel's problem This second example adds an elastic zone above the porous media. Such a modification is very useful for practical problems. Specifically, we deal now with a problem in non-overlapping domains. The configuration is inspired by Girault et al. [56]. This is considered as a new benchmark configuration that is prototypical for reservoir simulations with some overburden. In the same manner, an underburdon could have been added in addition. This leads, however, to no additional difficulties. This second example can be computed by further modifying the programming code or can be immediately run with another deal.II-based package DOpElib [62].

4.1.7.5.1 Configuration The computational domain is enlarged in height such that $[0, 100m] \times [0, 40m]$ and is sketched in Figure 9.

4.1.7.5.2 Boundary conditions As boundary conditions, we choose:

$$\begin{aligned} \sigma_B n &= \bar{t} \quad \text{on } \Gamma_{top}, \\ \sigma_B n &= 0 \quad \text{on } \Gamma_{r,S}, \\ p &= 0 \quad \text{on } \Gamma_{r,B}, \\ u_y &= 0 \quad \text{on } \Gamma_b, \\ u_x &= 0 \quad \text{on } \Gamma_{l,B} \cup \Gamma_{l,S}. \end{aligned}$$

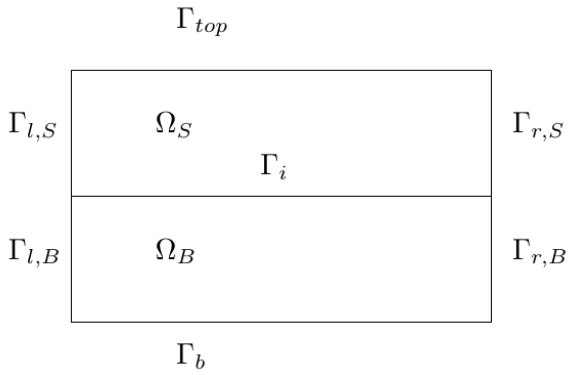


Figure 9: Configuration of augmented Mandel's problem.

4.1.7.5.3 Parameters We use the same Biot-region parameters as in the previous example. In addition, we use in the pure elastic zone the same Lamé coefficients, such that $\mu_B = \mu_S$ and $\lambda_B = \lambda_S$.

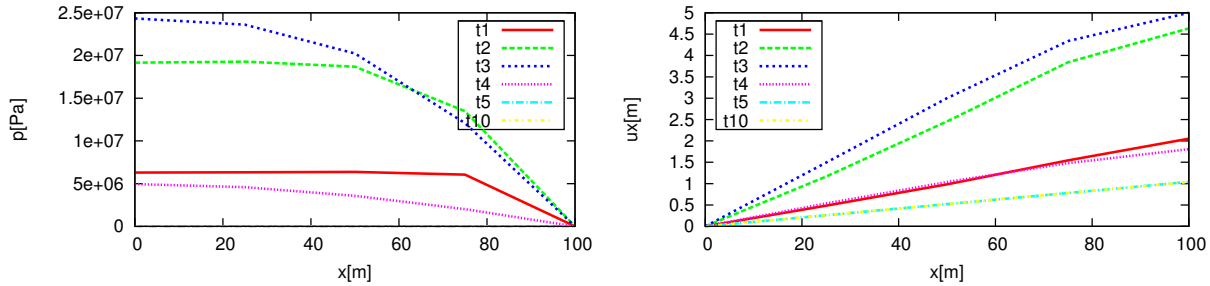


Figure 10: Augmented Mandel's problem: Pressure trace on the x-axis.

4.1.7.5.4 Discussion of our findings The augmented Mandel problem shows also the well-known Mandel Creyer effect as illustrated in Figure 10. Here, it is important to carefully implement the interface conditions on Γ_i. Otherwise the results are not correct from physical point of view. Graphical solutions of the surfaces of pressure distribution is shown in Figure 11. Here, we detect the typical pressure behavior on the x-axis.

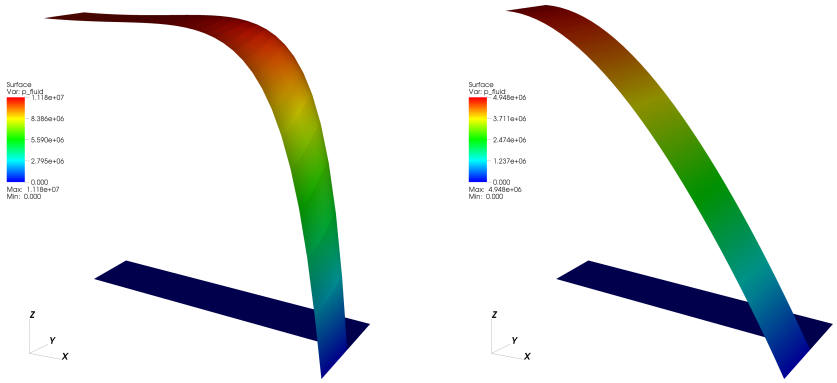


Figure 11: Augmented Mandel's problem: Pressure surface for two time steps t_1 and t_4.

4.2 Intermediate brief summary: several PDEs in one view

4.2.1 Poisson

Find $u : \Omega \to \mathbb{R}$:

$$-\Delta u = f \tag{14}$$

Properties: linear, stationary, scalar-valued.

4.2.2 Heat equation

Find $u : \Omega \times I \to \mathbb{R}$:

$$\partial_t u - \Delta u = f \tag{15}$$

Properties: linear, time-dependent, scalar-valued.

4.2.3 Wave equation

Find $U : \Omega \times I \to \mathbb{R}$:

$$\partial_t^2 u - \Delta u = f \tag{16}$$

Properties: linear, time-dependent, scalar-valued.

4.2.4 Biharmonic

Find $u : \Omega \to \mathbb{R}$:

$$\Delta^2 u = f \tag{17}$$

Properties: 4th order, linear, stationary, scalar-valued.

4.2.5 Eigenvalue

Find $u : \Omega \to \mathbb{R}$:

$$-\Delta u = \lambda u \tag{18}$$

Properties: linear, stationary, scalar-valued.

4.2.6 Linear transport

Find $u : \Omega \to \mathbb{R}$:

$$\partial_t u + b\nabla u = 0 \tag{19}$$

Properties: linear, nonstationary, scalar-valued.

4.2.7 Linearized elasticity

Find $u : \Omega \to \mathbb{R}^n$:

$$-\nabla \cdot \sigma(u) = f \tag{20}$$

with $\sigma = 2\mu e(u) + \lambda tr(e(u))I$ and $e(u) = \frac{1}{2}(\nabla u + \nabla u^T)$. Properties: linear, stationary, vector-valued.

4.3 Three important linear PDEs

From the previous considerations, we can extract **three important types of linear PDEs**. But we refrain from giving the impression that all differential equations can be classified and then a recipe for solving them applies. This is definitely not true - in particular for nonlinear PDEs. However, often 'new' equations can be related or simplified to these three types. For this reason, it is important to known them.

They read:

- Poisson problem: $-\Delta u = f$ is elliptic: second order in space and no time dependence.
- Heat equation: $\partial_t u - \Delta u = f$ is parabolic: second order in space and first order in time.
- Wave equation: $\partial_t^2 u - \Delta u = f$ is hyperbolic: second order in space and second order in time.

All these three equations have in common that their spatial order is two (the highest derivative with respect to spatial variables that occurs in the equation). With regard to time, there are differences: the heat equation is of first order whereas the wave equation is second order in time.

Let us now consider these three types in a bit more detail.

4.3.1 Elliptic equations and Poisson problem/Laplace equation

The Poisson equation plus boundary conditions is a boundary-value problem and will be derived from the general definition (very similar to the form before). Elliptic problems are second-order in space and have no time dependence.

Formulation 4.22. *Let $f : \Omega \to \mathbb{R}$ be given. Furthermore, Ω is an open, bounded set of $\mathbb{R}^n$. We seek the unknown function $u : \bar{\Omega} \to \mathbb{R}$ such that*

$$Lu = f \quad \text{in } \Omega, \tag{21}$$
$$u = 0 \quad \text{on } \partial\Omega. \tag{22}$$

Here, the linear second-order differential operator is defined by (divergence form):

$$Lu := -\sum_{i,j=1}^{n} \partial_{x_j}(a_{ij}(x)\partial_{x_i} u) + \sum_{i=1}^{n} b_i(x)\partial_{x_i} u + c(x)u, \quad u = u(x), \tag{23}$$

with the symmetry assumption $a_{ij} = a_{ji}$ and given coefficient functions a_{ij}, b_i, c. Alternatively we often use the compact notation with derivatives defined in terms of the nabla-operator:

$$Lu := -\nabla \cdot (a\nabla u) + b\nabla u + cu.$$

In non-divergence form the above differential operator reads:

$$Lu := -\sum_{i,j=1}^{n} a_{ij}(x)\partial_{x_j}\partial_{x_i} u + \sum_{i=1}^{n} b_i(x)\partial_{x_i} u + c(x)u, \quad u = u(x), \tag{24}$$

If the coefficients a_{ij} are C^1 functions, then both formulations can be rewritten into the other.

Remark 4.23. *Both versions have some advantages: the divergence form is more natural for energy methods and formulating weak formulations (basis of Galerkin finite elements), while the non-divergence form will be just used below for some theoretical aspects.*

*Finally we notice that the boundary condition (22) is called **homogeneous Dirichlet condition**.*

Remark 4.24. *Other possible boundary conditions are introduced in Section 4.5.*

Remark 4.25. *If the coefficient functions a, b, c also depend on the solution u we obtain a nonlinear PDE.*

Consider now the main part (from the non-divergence form):

$$\tilde{L}u := \sum_{i,j=1}^{n} a_{ij}(x)\partial_{ij}u$$

and we assume symmetry $a_{ij} = a_{ji}$. The operator $\tilde{L}$ (and also L of course) generates a quadratic form Σ, which is defined as

$$\Sigma(\xi) := \sum_{i,j=1}^{n} a_{ij}(x)\xi_i\xi_j.$$

The properties of the form Σ (and consequently the classification of the underlying PDE) depends on the eigenvalues of the matrix A:

$$A = (a_{ij})_{ij=1}^{n}.$$

Definition 4.26 (Elliptic: sign of all eigenvalues is the same)**.** *At the a given point $x \in \Omega$, the differential operator L is said to be elliptic if all eigenvalues (for the definition see Section 3.8.1) of A are non-zero and have the same sign. This is the same as saying A is symmetric, positive definite.*

Equivalently one can say:

Definition 4.27. *A PDE operator L is (uniformly) elliptic if there exists a constant $\theta > 0$ such that*

$$\sum_{i,j=1}^{n} a_{ij}(x)\xi_i\xi_j \geq \theta|\xi|^2,$$

for a.e. $x \in \Omega$ and all $\xi \in \mathbb{R}^n$.

Formulation 4.28 (Poisson problem)**.** *Setting in Formulation 4.22, $a_{ij} = \delta_{ij}$ and $b_i = 0$ and $c = 0$, we obtain the Laplace operator. Let $f : \Omega \to \mathbb{R}$ be given. Furthermore, Ω is an open, bounded set of $\mathbb{R}^n$. We seek the unknown function $u : \bar{\Omega} \to \mathbb{R}$ such that*

$$Lu = f \quad \text{in } \Omega, \tag{25}$$
$$u = 0 \quad \text{on } \partial\Omega. \tag{26}$$

Here, the linear second-order differential operator is defined by:

$$Lu := -\nabla \cdot (\nabla u) = -\Delta u.$$

Some important physical laws are related to the Laplace operator (partially taken from [51]):

1. Fick's law of chemical diffusion
2. Fourier's law of heat conduction
3. Ohm's law of electrical conduction
4. Small deformations in elasticity (e.g., membrane, clothesline).

In the following we discuss some characteristics of the solution of the Laplace problem, which can be generalized to general elliptic problems. Solutions to Poisson's equation attain their maximum on the boundary and not in the interior of the domain. This property will also carry over to the discrete problem and can be used to **check the correctness of the corresponding numerical solution.**

Definition 4.29 (Harmonic function)**.** *A C^2 function satisfying $\Delta u = 0$ is called a harmonic function.*

Theorem 4.30 (Strong maximum principle for the Laplace problem)**.** *Suppose $u \in C^2(\Omega) \cap C(\bar{\Omega})$ is a harmonic function. Then*

$$\max_{\bar{\Omega}} u = \max_{\partial\Omega} u.$$

Moreover, if Ω is connected and there exists a point $y \in \Omega$ in which

$$u(y) = \max_{\bar{\Omega}} u,$$

then u is constant within Ω. The same holds for $-u$, but then for minima.

Proof. See Evans [51]. □

Remark 4.31 (Maximum principle in practice)**.** *An illustration of the maximum principle in practice for a 1D setting is provided in Figure 13, where the maximum values are clearly obtained on the two boundary points of the domain.*

Corollary 4.32. *For finding $u \in C^2(\Omega) \cap C(\bar{\Omega})$, such that*

$$\begin{aligned} \Delta u &= 0 \quad \text{in } \Omega, \\ u &= g \quad \text{on } \partial\Omega, \end{aligned}$$

with $g \geq 0$, it holds that

$$u > 0 \quad \text{everywhere in } \Omega \quad \text{if} \quad g > 0 \text{ on some part of } \partial\Omega. \tag{27}$$

This property also shows that the Laplace equation has an infinite propagation speed of perturbations: if we change the solution by ε in one point, the solution in the entire domain will (slightly) change.

From the maximum principle we obtain immediately uniqueness of a solution:

Theorem 4.33 (Uniqueness of the Laplace problem)**.** *Let $g \in C(\partial\Omega)$ and $f \in C(\Omega)$. Then there exists at most one solution $u \in C^2(\Omega) \cap C(\bar{\Omega})$ of the boundary-value problem:*

$$-\Delta u = f \quad \text{in } \Omega, \tag{28}$$
$$u = g \quad \text{on } \partial\Omega. \tag{29}$$

Proof. If u_1 and u_2 both satisfy the equation, we apply the maximum principle to the harmonic functions

$$w = u_1 - u_2, \quad w = -(u_1 - u_2).$$

This yields $u_1 = u_2$. □

4.3.2 Parabolic equations and heat equation

We now study parabolic problems, which contain as the most famous example the heat equation. Parabolic problems are second order in space and first order in time.

In this section, we assume $\Omega \subset \mathbb{R}^n$ to be an open and bounded set. The time interval is given by $I := (0, T]$ for some fixed end time value $T > 0$.

We consider the initial/boundary-value problem (IBVP):

Formulation 4.34. *Let $f : \Omega \times I \to \mathbb{R}$ and $u_0 : \Omega \to \mathbb{R}$ be given. We seek the unknown function $u : \bar{\Omega} \times I \to \mathbb{R}$ such that*[2]

$$\partial_t u + Lu = f \quad \text{in } \Omega \times I, \tag{30}$$
$$u = 0 \quad \text{on } \partial\Omega \times [0, T], \tag{31}$$
$$u = u_0 \quad \text{on } \Omega \times \{t = 0\}. \tag{32}$$

[2]For the heat equation, we should have better used the notation T, the temperature, for the variable rather than u. But to stay consistant with the entire notation, we still use u.

Here, the linear second-order differential operator is defined by:

$$Lu := -\sum_{i,j=1}^{n} \partial_{x_j}(a_{ij}(x,t)\partial_{x_i}u) + \sum_{i=1}^{n} b_i(x,t)\partial_{x_i}u + c(x,t)u, \quad u = u(x,t)$$

for given (possibly spatial and time-dependent) coefficient functions a_{ij}, b_i, c.

We have the following:

Definition 4.35. *The PDE operator $\partial_t + L$ is (uniformly) parabolic if there exists a constant $\theta > 0$ such that*

$$\sum_{i,j=1}^{n} a_{ij}(x,t)\xi_i\xi_j \geq \theta|\xi|^2,$$

for all $(x,t) \in \Omega \times I$ and all $\xi \in \mathbb{R}^n$. In particular this operator is elliptic in the spatial variable x for each fixed time $0 \leq t \leq T$.

Definition 4.36 (Parabolic: one eigenvalue zero; all others have the same sign)**.** *For parabolic PDEs one eigenvalue of the space-time coefficient matrix is zero whereas the others have the same sign.*

Formulation 4.37 (Heat equation)**.** *Setting in Formulation 4.34, $a_{ij} = \delta_{ij}$ and $b_i = 0$ and $c = 0$, we obtain the Laplace operator. Let $f : \Omega \to \mathbb{R}$ be given. Furthermore, Ω is an open, bounded set of $\mathbb{R}^n$. We seek the unknown function $u : \bar{\Omega} \to \mathbb{R}$ such that*

$$\partial_t u + Lu = f \quad \text{in } \Omega \times I, \tag{33}$$
$$u = 0 \quad \text{on } \partial\Omega \times [0,T], \tag{34}$$
$$u = u_0 \quad \text{on } \Omega \times \{t = 0\}. \tag{35}$$

Here, the linear second-order differential operator is defined by:

$$Lu := -\nabla \cdot (\nabla u) = -\Delta u.$$

The heat equation has an infinite propagation speed for disturbances. If the initial temperature is non-negative and is positive somewhere, the temperature at any later time is everywhere positive. For the heat equation, a very similar maximum principle as for elliptic problems holds true. This principle is extended to the parabolic boundary. We deal with a diffusions problem and flow points into the direction of the negative gradient. Define $Q = (0,T] \times \Omega$ the space-time cylinder. The parabolic boundary is given by $\Gamma := \bar{Q} - Q$, namely:

$$\Gamma = \Big([0,T] \times \partial\Omega\Big) \cup \Big(\{0\} \times \Omega\Big).$$

This definition contains the bottom and the vertical sides of $Q \times [0,T]$, but not the top. For $f = 0$ the maximum temperature u is obtained on the parabolic boundary. In other words, the maximum is taken at the initial time step or on the boundary of the spatial domain.

Theorem 4.38 (Strong parabolic maximum principle for the heat equation)**.** *Suppose $u \in C_1^2(\Omega \times I) \cap C(\bar{\Omega} \times I)$ is a solution of the heat equation:*

$$\partial_t u = \Delta u \quad \text{in } \Omega \times I = (0,T).$$

Then

$$\max_{\bar{\Omega} \times I} u = \max_{\Gamma} u.$$

In other words: the maximum u (or the minimum) is obtained either at $t = 0$ or on $[0,T] \times \partial\Omega$. A numerical test demonstrating the maximum principle is provided in Section 12.3.12.

Remark 4.39. *As for elliptic problems, the maximum principle can be used to proof uniqueness for the parabolic case.*

4.3.3 Hyperbolic equations and the wave equation

In this section, we shall study hyperbolic problems, which are natural generalizations of the wave equation. As for parabolic problems, let $\Omega \subset \mathbb{R}^n$ to be an open and bounded set. The time interval is given by $I := (0, T]$ for some fixed end time value $T > 0$.

We consider the initial/boundary-value problem:

Formulation 4.40. *Let $f : \Omega \times I \to \mathbb{R}$ and $u_0, v_0 : \Omega \to \mathbb{R}$ be given. We seek the unknown function $u : \bar{\Omega} \times I \to \mathbb{R}$ such that*

$$\partial_t^2 u + Lu = f \quad \text{in } \Omega \times I, \tag{36}$$
$$u = 0 \quad \text{on } \partial\Omega \times [0, T], \tag{37}$$
$$u = u_0 \quad \text{on } \Omega \times \{t = 0\}, \tag{38}$$
$$\partial_t u = v_0 \quad \text{on } \Omega \times \{t = 0\}. \tag{39}$$

In the last line, $\partial_t u = v$ can be identified as the velocity. Furthermore, the linear second-order differential operator is defined by:

$$Lu := -\sum_{i,j=1}^{n} \partial_{x_j}(a_{ij}(x,t)\partial_{x_i} u) + \sum_{i=1}^{n} b_i(x,t)\partial_{x_i} u + c(x,t)u, \quad u = u(x,t)$$

for given (possibly spatial and time-dependent) coefficient functions a_{ij}, b_i, c.

Remark 4.41. *The wave equation is often written in terms of a first-order system in which the velocity is introduced and a second-order time derivative is avoided. Then the previous equation reads: Find $u : \bar{\Omega} \times I \to \mathbb{R}$ and $v : \bar{\Omega} \times I \to \mathbb{R}$ such that*

$$\partial_t v + Lu = f \quad \text{in } \Omega \times I, \tag{40}$$
$$\partial_t u = v \quad \text{in } \Omega \times I, \tag{41}$$
$$u = 0 \quad \text{on } \partial\Omega \times [0, T], \tag{42}$$
$$u = u_0 \quad \text{on } \Omega \times \{t = 0\}, \tag{43}$$
$$v = v_0 \quad \text{on } \Omega \times \{t = 0\}. \tag{44}$$

We have the following:

Definition 4.42. *The PDE operator $\partial_t^2 + L$ is (uniformly) hyperbolic if there exists a constant $\theta > 0$ such that*

$$\sum_{i,j=1}^{n} a_{ij}(x,t)\xi_i\xi_j \geq \theta|\xi|^2,$$

for all $(x,t) \in \Omega \times I$ and all $\xi \in \mathbb{R}^n$. In particular this operator is elliptic in the spatial variable x for each fixed time $0 \leq t \leq T$.

Definition 4.43 (Hyperbolic: one eigenvalue has a different sign than all others)**.** *A space-time differential operator is called hyperbolic when one eigenvalue has a different sign than all the others.*

And as before, setting the coefficient functions to trivial values, we obtain the original wave equation. In contrast to parabolic problems, a strong maximum principle does not hold for hyperbolic equations. And consequently, the propagation speed is finite. This means that a disturbance at some point $x \in \Omega$ at some time $t \in I$ is not immediately transported to any point $x \in \Omega$ at any time $t \in I$.

4.4 Nonconstant coefficients. Change of the PDE types elliptic, parabolic, hyperbolic

Even so that we define separate types of PDEs, in many processes there is a mixture of these classes in one single PDE - depending on the size of certain parameters. For instance, the Navier-Stokes equations (only showing here the conservation of momentum):

$$\partial_t v + v \cdot \nabla v - \frac{1}{Re}\Delta v + \nabla p = f$$

for modeling fluid flow, vary between parabolic and hyperbolic type depending on the Reynold's number $Re \sim \nu_f^{-1}$ (respectively a characteristic velocity and a characteristic length). For $Re \to 0$, we obtain a first-order hyperbolic equation:

$$\partial_t v + v \cdot \nabla v + \nabla p = f,$$

while for $Re \to \infty$, we obtain a parabolic type:

$$\partial_t v - \frac{1}{Re}\Delta v + \nabla p = f.$$

Relating this to the general definition of the differential operator L, we can say that the coefficients a_{ij} of the differential operator L are non-constant and change with respect to space and time.

4.5 Boundary conditions and initial data

As seen in the previous sections, all PDEs are complemented with boundary conditions and in time-dependent cases, also with initial conditions. Actually, these are crucial ingredients for solving differential equations. Often, one has the (wrong) impression that only the PDE itself is of importance. But what happens on the boundaries finally yields the 'result' of the computation. And this holds true for analytical (classical) as well as computational solutions.

4.5.1 Boundary conditions

In Section 4.3, for simplicity, we have mainly dealt with homogeneous Dirichlet conditions of the form $u = 0$ on the boundary $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N \cup \partial\Omega_R$. In general three types of conditions are of importance:

- Dirichlet (or essential) boundary conditions: $u = g_D$ on $\partial\Omega_D$; when $g_D = 0$ we say 'homogeneous' boundary condition.
- Neumann (or natural) boundary conditions: $\partial_n u = g_N$ on $\partial\Omega_N$; when $g_N = 0$ we say 'homogeneous' boundary condition.
- Robin (third type) boundary condition: $au + b\partial_n u = g_R$ on $\partial\Omega_R$; when $g_R = 0$ we say 'homogeneous' boundary condition.

On each boundary section, only one of the three conditions can be described. In particular the natural conditions generalize when dealing with more general operators L. Here in this section, we have assumed $L := -\Delta u$.

Remark 4.44. *Going from differential equations to variational formulations, Dirichlet conditions are explicitely built into the function space, for this reason they are called* ***essential boundary conditions****. Neumann conditions appear explicitly through integration by parts; for this reason they are called* ***natural boundary conditions****.*

4.5.2 Initial conditions

The number of initial conditions depends on the order of temporal time derivatives. The usual notation is:

$$u = u_0 \quad \text{in } \Omega \times \{t = 0\}.$$

As examples, we have:

- $\partial_t u - ...$: first order in time: one initial condition.
- $\partial_t^2 u - ...$: second order in time: two initial conditions.

4.5.3 Example: temperature in a room

Example 4.45. *Let us compute the heat distribution in our room b302. The room volume is Ω. The window wall is a Dirichlet boundary $\partial_D\Omega$ and the remaining walls are Neumann boundaries $\partial_N\Omega$. Let K be the air viscosity.*

We consider the heat equation: Find $T : \Omega \times I \to \mathbb{R}$ such that

$$\begin{aligned} \partial_t T + (v \cdot \nabla)T - \nabla \cdot (K\nabla T) &= f && \text{in } \Omega \times I, \\ T &= 18°C && \text{on } \partial_D\Omega \times I, \\ K\nabla T \cdot n &= 0 && \text{on } \partial_N\Omega \times I, \\ T(0) &= 15°C && \text{in } \Omega \times \{0\}. \end{aligned}$$

The homogeneous Neumann condition means that there is no heat exchange on the respective walls (thus neighboring rooms will have the same room temperature on the respective walls). The nonhomogeneous Dirichlet condition states that there is a given temperature of $18°C$, which is constant in time and space (but this condition may be also non-constant in time and space). Possible heaters in the room can be modeled via the right hand side f. The vector $v : \Omega \to \mathbb{R}^3$ denotes a given flow field yielding a convection of the heat, for instance wind. We can assume $v \approx 0$. Then the above equation is reduced to the original heat equation: $\partial_t T - \nabla \cdot (K\nabla T) = f$.

4.6 The general definition of a differential equation

After the previous examples, let us precise the definition of a differential equation. In order to allow for largest possible generality we first need to introduce some notation. Common is to use the **multiindex notation** as introduced in Section 3.6.

4.6.1 Single-valued differential equations

Definition 4.46 (Evans [51], single-valued PDE)**.** *Let $\Omega \subset \mathbb{R}^n$ be open. Here, n denotes the total dimension including time. Furthermore, let $k \geq 1$ an integer that denotes the order of the differential equation. Then, a differential equation can be expressed as: Find $u : \Omega \to \mathbb{R}$ such that*

$$F(D^k u, D^{k-1}u, \dots, D^2 u, Du, u, x) = 0 \quad x \in \Omega,$$

where

$$F : \mathbb{R}^{n^k} \times \mathbb{R}^{n^{k-1}} \times \dots \times \mathbb{R}^{n^2} \times \mathbb{R}^n \times \mathbb{R} \times \Omega \to \mathbb{R}.$$

Example 4.47. *We provide some examples. Let us assume the spatial dimension to be 2 and the temporal dimension is 1. That is for a time-dependent ODE (ordinary differential equation) problem $n = 1$ and for time-dependent PDE cases, $n = 2 + 1 = 3$, and for stationary PDE examples $n = 2$.*

1. *ODE model problem: $F(Du, u) := u' - au = 0$ where $F : \mathbb{R}^1 \times \mathbb{R} \to \mathbb{R}$. Here $k = 1$.*
2. *Laplace operator: $F(D^2u) := -\Delta u = 0$ where $F : \mathbb{R}^4 \to \mathbb{R}$. That is $k = 2$ and lower derivatives of order 1 and 0 do not exist. We notice that in the general form it holds*

   $$D^2 u = \begin{pmatrix} \partial_{xx}u & \partial_{yx}u \\ \partial_{xy}u & \partial_{yy}u \end{pmatrix}$$

   and specficially now for the Laplacian (consider the sign as well!):

   $$D^2 u = \begin{pmatrix} -\partial_{xx}u & 0 \\ 0 & -\partial_{yy}u \end{pmatrix}$$

3. *Heat equation: $F(D^2u, Du) = \partial_t u - \Delta u = 0$ where $F : \mathbb{R}^9 \times \mathbb{R}^3 \to \mathbb{R}$. Here $k = 2$ is the same as before, but a lower-order derivative of order 1 in form of the time derivative does exist. We provide again details on D^2u:*

   $$D^2 u = \begin{pmatrix} \partial_{tt}u & \partial_{xt}u & \partial_{yt}u \\ \partial_{tx}u & \partial_{xx}u & \partial_{yx}u \\ \partial_{ty}u & \partial_{xy}u & \partial_{yy}u \end{pmatrix}, \qquad Du = \begin{pmatrix} \partial_t u \\ \partial_x u \\ \partial_y u \end{pmatrix}$$

and for the heat equation:

$$D^2u = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -\partial_{xx}u & 0 \\ 0 & 0 & -\partial_{yy}u \end{pmatrix}, \qquad Du = \begin{pmatrix} \partial_t u \\ 0 \\ 0 \end{pmatrix}$$

4. *Wave equation:* $F(D^2u) = \partial_t^2 u - \Delta u = 0$ *where* $F : \mathbb{R}^9 \to \mathbb{R}$. *Here it holds specifically*

$$D^2u = \begin{pmatrix} \partial_{tt}u & 0 & 0 \\ 0 & -\partial_{xx}u & 0 \\ 0 & 0 & -\partial_{yy}u \end{pmatrix}.$$

We finally compare the coeffcient matrices and recall the type classifications. Let us denote the coefficient matrix by A. Then:

$$D^2u = \begin{pmatrix} -\partial_{xx}u & 0 \\ 0 & -\partial_{yy}u \end{pmatrix} \quad \to \quad A = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \quad \to \quad \textit{Same sign: elliptic}$$

$$D^2u = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -\partial_{xx}u & 0 \\ 0 & 0 & -\partial_{yy}u \end{pmatrix} \quad \to \quad A = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \quad \to \quad \textit{One eigenvalue zero; others same sign: parabolic}$$

$$D^2u = \begin{pmatrix} \partial_{tt} & 0 & 0 \\ 0 & -\partial_{xx}u & 0 \\ 0 & 0 & -\partial_{yy}u \end{pmatrix} \quad \to \quad A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \quad \to \quad \textit{One eigenvalue has different sign: hyperbolic}$$

4.6.2 Vector-valued differential equations, PDE systems

Definition 4.48 (Evans [51], PDE system)**.** *Let* $\Omega \subset \mathbb{R}^n$ *be open. Here,* n *denotes the total dimension including time. Furthermore, let* $k \geq 1$ *an integer that denotes the order of the differential equation. Then, a vector-valued differential equation can be expressed as: Find* $u = (u_1, \ldots, u_m) : \Omega \to \mathbb{R}^m$ *such that*

$$F(D^k u, D^{k-1}u, \ldots, D^2u, Du, u, x) = 0 \quad x \in \Omega,$$

where

$$F : \mathbb{R}^{mn^k} \times \mathbb{R}^{mn^{k-1}} \times \ldots \times \mathbb{R}^{mn^2} \times \mathbb{R}^{mn} \times \mathbb{R}^m \times \Omega \to \mathbb{R}^m.$$

Example 4.49. *We discuss two vector-valued PDEs:*

1. *Vector-Laplace:*

$$-\Delta u = -\Delta \begin{pmatrix} u_1 \\ \vdots \\ u_m \end{pmatrix} = \begin{pmatrix} f_1 \\ \vdots \\ f_m \end{pmatrix}.$$

Say in $n = 2$, *and* $m = 3$ *we have*

$$-\Delta u = -\Delta \begin{pmatrix} u_1(x,y) \\ u_2(x,y) \\ u_3(x,y) \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix}.$$

Here,

$$F(D^2u) := -\Delta u - f,$$

with

$$F : \mathbb{R}^{3 \cdot 2^2} \times \Omega \to \mathbb{R}^3.$$

2. *Linearized elasticity (see also Chapter 11): Find* $u = (u_1, u_2, u_3) : \Omega \to \mathbb{R}^3$ *such that*

$$-\nabla \cdot \sigma(u) = f,$$

where $f = (f_1, f_2, f_3)$ *and the (very simple!) stress tensor*

$$\sigma(u) = \frac{1}{2}(\nabla u + \nabla u^T).$$

In comparison to the previous vector Laplace, we have now cross terms in the off-diagonal blocks:

$$\sigma(u) = \frac{1}{2}\left(\begin{pmatrix} \partial_x u_x & \partial_y u_x & \partial_z u_x \\ \partial_x u_y & \partial_y u_y & \partial_z u_y \\ \partial_x u_z & \partial_y u_z & \partial_z u_z \end{pmatrix} + \begin{pmatrix} \partial_x u_x & \partial_x u_y & \partial_x u_z \\ \partial_y u_x & \partial_y u_y & \partial_y u_z \\ \partial_z u_x & \partial_z u_y & \partial_z u_z \end{pmatrix}\right)$$

Here,

$$F(D^2 u) := -\nabla \cdot \sigma(u) - f,$$

with (as for vector-Laplace)

$$F : \mathbb{R}^{3 \cdot 2^2} \times \Omega \to \mathbb{R}^3.$$

4.7 Classifications into linear and nonlinear PDEs

Based on the previous notation, we now provide classifications of **linear** and **nonlinear** differential equations. Simply speaking: each differential equation, which is not linear is called nonlinear. However, in the nonlinear case a further refined classification can be undertaken.

Definition 4.50 (Evans[51])**.** *Differential equations are divided into linear and nonlinear classes as follows:*

1. *A differential equation is called* ***linear*** *if it is of the form:*

$$\sum_{|\alpha| \le k} a_\alpha(x) D^\alpha u - f(x) = 0.$$

2. *A differential equation is called* ***semi-linear*** *if it is of the form:*

$$\sum_{|\alpha| = k} a_\alpha(x) D^\alpha u + a_0(D^{k-1}u, \dots, D^2 u, Du, u, x) = 0.$$

Here, nonlinearities may appear in all terms of order $|\alpha| < k$*, but the highest order* $|\alpha| = k$ *is fully linear.*

3. *A differential equation is called* ***quasi-linear*** *if it is of the form:*

$$\sum_{|\alpha| = k} a_\alpha(D^{k-1}u, \dots, D^2 u, Du, u, x) D^\alpha u + a_0(D^{k-1}u, \dots, D^2 u, Du, u, x) = 0.$$

Here, full nonlinearities may appear in all terms of order $|\alpha| < k$*, in the highest order* $|\alpha| = k$*, nonlinear terms appear up to order* $|\alpha| < k$*.*

4. *If none of the previous cases applies, a differential equation is called (fully)* ***nonlinear****.*

Remark 4.51. *In theory as well as practice it holds: the more nonlinear and equation is, the more difficult the analysis becomes and the more challenging is the design of good numerical methods.*

4.7.1 Examples

How can we now proof in practice, whether a PDE is linear or nonlinear? The simplest way is to go term by term and check the linearity condition from Definition 3.15.

Example 4.52. *We provide again some examples:*

1. *All differential equations from Example 4.47 are* ***linear****. Check term by term Definition 3.15!*

2. *Euler equations (fluid dynamics, special case of Navier-Stokes with zero viscosity). Here,* $n = 2+1+1 = 4$ *(in two spatial dimensions) in which we have* 2 *velocity components,* 1 *pressure variable, and* 1 *temporal variable. Let us consider the momentum part of the Euler equations:*

$$\partial_t v_f + v_f \cdot \nabla v_f + \nabla p_f = f.$$

   Here the highest order is $k = 1$ *(in the temporal variable as well as the spatial variable). But in front of the spatial derivative, we multiply with the zero-order term* v_f*. Consequently, the Euler equations are* ***quasi-linear*** *because a lower-order term of the solution variable is multiplied with the highest derivative.*

3. *Navier-Stokes momentum equation:*

$$\partial_t v_f - \rho_f \nu_f \Delta v_f + v_f \cdot \nabla v_f + \nabla p_f = f.$$

   Here $k = 2$*. But the coefficients in front of the highest order term, the Laplacian, do not depend on* v_f*. Consequently, the Navier-Stokes equations are neither fully nonlinear nor quasi-linear. Well, we have again the nonlinearity of the first order convection term* $v_f \cdot \nabla v_f$*. Thus the Navier-Stokes equations are* ***semi-linear****.*

4. *A fully* ***nonlinear*** *situation would be:*

$$\partial_t v_f - \rho_f \nu_f (\Delta v_f)^2 + v_f \cdot \nabla v_f + \nabla p_f = f.$$

Example 4.53 (Development of numerical methods for nonlinear equations)**.** *In case you are given a nonlinear IBVP (initial-boundary value problem) and want to start developing numerical methods for this specific PDE, it is often much easier to start with appropriate simplifications in order to build and analyze step-by-step your final method. Let us say you are given the nonlinear time-dependent PDE*

$$\nabla u \partial_t^2 u + u \cdot \nabla u - (\Delta u)^2 = f$$

Then, you could tackle the problem as follows:

1. *Consider the linear equation:*

$$\partial_t^2 u - \Delta u = f$$

   which is nothing else than the wave equation.

2. *Add a slight nonlinearity to make the problem semi-linear:*

$$\partial_t^2 u + u \cdot \nabla u - \Delta u = f$$

3. *Add* ∇u *such that the problem becomes quasi-linear:*

$$\nabla u \partial_t^2 u + u \cdot \nabla u - \Delta u = f$$

4. *Make the problem fully nonlinear by considering* $(\Delta u)^2$*:*

$$\nabla u \partial_t^2 u + u \cdot \nabla u - (\Delta u)^2 = f.$$

In each step, make sure that the corresponding numerical solution makes sense and that your developments so far are correct. If yes, proceed to the next step.

Remark 4.54. *The contrary to the previous example works as well and should be kept in mind! If you have implemented the full nonlinear PDE and recognize a difficulty, you can reduce the PDE term by term and make it step by step simpler. The same procedure holds true for the mathematical analysis.*

Exercise 3. *Classify the following PDEs into linear and nonlinear PDEs (including a short justification):*

$$\begin{aligned} -\nabla \cdot (|\nabla u|^{p-2}\nabla u) &= f \\ det(\nabla^2 u) &= f \\ \partial_t^2 u + 2a\partial_t u - \partial_{xx} u = 0, \quad a &> 0 \\ \partial_t u + u\partial_x u &= 0 \\ \partial_{tt} - \Delta u + m^2 u = 0, \quad m &> 0. \end{aligned}$$

How can we linearize nonlinear PDEs? Which terms need to be simplified such that we obtain a linear PDE?

4.8 Further remarks on the classification into three important types

The previous general definitions into *elliptic, parabolic, and hyperbolic* types can be made clearer when we simplify the situation of (23) in $\mathbb{R}^2$:

$$Lu := a_{11}\partial_x^2 u + 2a_{12}\partial_x\partial_y u + a_{22}\partial_y^2 u + a_1\partial_x u + a_2\partial_y u + au = f$$

with, for the moment, constant coefficients a_{ij}. Moreover, a_{11}, a_{12}, a_{22} should not be zero simultaneously. The main part of L can be represented by the quadratic form:

$$q(x, y) = a_{11}x^2 + 2a_{12}xy + a_{22}y^2.$$

Solving $q(x, y) = 0$ yields conic sections in the xy plane. Specifically, we have a relation to geometry:

- Circle, ellipse: $x^2 + y^2 = 1$;
- Parabola: $y = x^2$;
- Hyperbola: $x^2 - y^2 = 1$.

In order to apply this idea to L we write the main part of L in matrix-vector form:

$$L_0 := a_{11}\partial_x^2 + 2a_{12}\partial_x\partial_y + a_{22}\partial_y^2 = \begin{pmatrix} \partial_x \\ \partial_y \end{pmatrix}^T \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} \partial_x \\ \partial_y \end{pmatrix}.$$

The type of the PDE can now be dermined by investigating the properties of the coefficient matrix:

Proposition 4.55. *The differential operator L can be brought through a linear transformation to one of the following forms:*

1. Elliptic, *if* $a_{12}^2 < a_{11}a_{22}$.
   $\to \partial_x^2 u + \partial_y^2 u + \ldots = 0 \quad \overset{example}{\to} \Delta u = 0$
2. Parabolic, *if* $a_{12}^2 = a_{11}a_{22}$.
   $\to \partial_x^2 u + \ldots = 0 \quad \overset{example}{\to} \partial_t u - \Delta u = 0$
3. Hyperbolic, *if* $a_{12}^2 > a_{11}a_{22}$.
   $\to \partial_x^2 u - \partial_y^2 u + \ldots = 0 \quad \overset{example}{\to} \partial_t^2 u - \Delta u = 0$

The $\ldots$ represent terms of lower order.

Remark 4.56. *Recall that this corresponds exactly to our previous definitions of elliptic, parabolic, and hyperbolic cases.*

Proof. We proof the result for the elliptic case. The other cases will follow in a similar manner. Without loss of generality, we assume $a_{11} = 1, a_1 = a_2 = a_0 = 0$. Completing the square yields

$$\begin{aligned} 0 &= (\partial_x + a_{12}\partial_y)^2 u + (a_{22} - a_{12}^2)\partial_y^2 u \\ &= (\partial_x^2 + 2a_{12}\partial_x\partial_y + a_{12}^2\partial_y^2)u + (a_{22} - a_{12}^2)\partial_y^2 u \end{aligned} \tag{45}$$

under the assumption that $a_{12}^2 < a_{11} \cdot a_{22}$. Set $b := \sqrt{a_{22} - a_{12}^2} > 0$.

We now do a coordinate transformation

$$x = \xi, \quad y = a_{12} \cdot \xi + b \cdot \eta \quad \text{(linear in } \xi \text{ and } \eta\text{)}$$

Recall how derivatives transform under coordinate transformations. We define

$$v(\xi, \eta) = u(x, y)$$

and interprete x and y as functions:

$$x(\xi, \eta), y(\xi, \eta) \to v(\xi, \eta) = u(x(\xi, \eta), y(\xi, \eta)).$$

We compute $(\partial_\xi v, \partial_\eta v)$:

$$\begin{aligned} (\partial_\xi v, \partial_\eta v) &= (\partial_x u, \partial_y u)\begin{pmatrix} \partial_\xi x & \partial_\eta x \\ \partial_\xi y & \partial_\eta y \end{pmatrix} \\ &= (\partial_x u, \partial_y u)\begin{pmatrix} 1 & 0 \\ a_{12} & b \end{pmatrix} \\ &= (\partial_x u + a_{12}\partial_y u, 0 \cdot \partial_x u + b \cdot \partial_y u). \end{aligned}$$

This yields: $\partial_\xi = \partial_x + a_{12}\partial_y$ and $\partial_\eta = b \cdot \partial_y$.

We plug the result into (45). The resulting equation is:

$$\partial_\xi^2 + \partial_\eta^2 = 0,$$

i.e., the Laplace equation.

□

Remark 4.57. *This definition also works for L that depends on space and time. Just replace the variable y through the time t.*

Example 4.58. *Classify the following equations:*

1. $\partial_{xx}u - 5\partial_{xy}u = 0$
2. $4\ \partial_{xx}u - 12\partial_{xy}u + 9\partial_{yy}u + \partial_y u = 0$
3. $4\ \partial_{xx}u + 6\partial_{xy}u + 9\partial_{yy}u = 0$

As previously discussed we compute the discriminant $D = a_{12}^2 - a_{11}a_{22}$ in order to see which case we have on hand:

1. $D = (-\frac{5}{2})^2 - 1 \cdot 0 = \frac{25}{4} > 0 \quad \to$ *hyperbolic*
2. $D = (-6)^2 - 4 \cdot 9 = 36 - 36 = 0 \quad \to$ *parabolic*
3. $D = 3^2 - 4 \cdot 9 = 9 - 36 = -25 < 0 \quad \to$ *elliptic.*

Example 4.59. *Determine subdomains of the xy plane in which*

$$y\partial_{xx}u - 2\partial_{xy}u + x\partial_{yy}u = 0$$

is elliptic, hyperbolic or parabolic. First, we have $D = (-1)^2 - y \cdot x = 1 - yx$. The equation is parabolic for $xy = 1$ because $1 - 1 = 0$. The elliptic parts are convex $xy > 1$ because $D = 1 - yx < 0$. In the connected part $xy < 1$, the equation is hyperbolic.

4.9 PDEs that are not elliptic, parabolic or hyperbolic

We have intensively discussed in the previous sections the classification into elliptic, parabolic, and hyperbolic types. It should be emphasized that these are **convenient** classifications for which well developed theories exist. But this does not mean that any linear PDE needs to fit into these types. For instance,

$$\partial_t u - \partial_{xx} u - \partial_{zz} u = f$$

is also a PDE that is very similar to a parabolic one, but the term $\partial_{yy} u$ is missing. Consequently, the coefficient matrix has two eigenvalues that are zero and two eigenvalues that are minus one. This is not simply a parabolic PDE anymore, but still a PDE that we can investigate and numerically model. Also, mixed derivatives might appear as well:

$$\partial_{tt}^2 u - \partial_{xx} u - \partial_{yz} u - \partial_{zx} u = f$$

This is a linear second order PDE, which is also not anymore parabolic or hyperbolic, but still a valid PDE that we can try to treat numerically. PDEs can also be of higher order:

$$\partial_{tt}^2 u + \Delta^2 u = f$$

which is a linear fourth-order PDE. Or

$$\partial_{tt}^2 u - \partial_{xxy} u = f,$$

which is a third-order PDE. All these PDEs in this section are mathematically valid, but we do not claim that they have specific physical applications though. However, fourth-order PDEs are well-known in mechanics for plates and beams; see [24].

4.10 Chapter summary and outlook

In this extended introduction, we formulated several important PDEs and discussed prototype applications and equations. Furthermore, we classified PDEs and introduced their level of linearity. In the next section, we first state some characteristic classifications of differential equations.

5 Classifications

In this chapter we classify PDEs into different categories:

Definition 5.1 (Classification of differential equations)**.** *The following properties are distinguished:*

1. *Independent variables $(x, y, z, t, \ldots)$; if only one, then ODE (ordinary differential equation), otherwise PDE;*
2. *Steady-state problems or time-dependent PDEs;*
3. *Physical conservation properties such as mass, momentum, diffusion, transport, energy;*
4. *Order of a PDE (in time, space, or both);*
5. *Single equations and PDE systems;*
6. *Nonlinear problems:*
   - *Nonlinearities in the PDE through state variables,*
   - *Nonlinear constitutive laws,*
   - *Inequality constraints;*
7. *Coupled systems:*
   - *coupling through model or material parameters,*
   - *linear or nonlinear coupling terms,*
   - *right-hand sides,*
   - *interfaces.*

5.1 Physical conservation properties

Despite not of main interest in these lecture notes, but rather in Numerical Methods for Continuum Mechanics, let us briefly comment on them. Often, conservation laws are useful to know in order to choose the correct numerical methods. One example is energy conservation (see also Chapter 12) for which time-stepping schemes with minimal numerical diffusion must be chosen.

5.2 Order of a PDE

As we have already seen in the previous chapters, the first property that we can distingish and which is quite important is the **order** of a PDE. Most problems have probably order 2 in space. Examples are the Laplacian, heat equation, wave equation, p-Laplace, Navier-Stokes. In time we deal mainly of order 0 (Poisson) to 2 (wave equation). In elasticity, several equations are of order 4 in space. Examples are beams.

5.3 Single equations versus PDE systems

Definition 5.2 (Single equation)**.** *A single PDE consists of determining one solution variable, e.g.,*

$$u : \Omega \subset \mathbb{R}^d \to \mathbb{R}.$$

Typical examples are Poisson's problem, the heat equation, wave equation, Monge-Ampère equation, Hamiliton-Jacobi equation, p-Laplacian.

Definition 5.3 (PDE system)**.** *A PDE system determines a solution vector*

$$u = (u_1, \ldots, u_d) : \Omega \to \mathbb{R}^d.$$

For each $u_i, i = 1, \ldots, d$, a PDE must be solved. Inside these PDEs, the solution variables may depend on each other or not. Typical examples of PDE systems are linearized elasticity, nonlinear elasto-dynamics, Maxwell's equations.

5.4 Linear versus nonlinear

Simply speaking: each differential equation, which is not linear is called nonlinear. However, in the nonlinear case a further refined classification can be undertaken as we have seen before.

When we deal with a nonlinear equation, we often in terms of the notation: Find $u \in V$ such that

$$a(u)(\phi) = l(\phi) \quad \forall \phi \in V.$$

In the so called semi-linear form $a(u)(\phi)$, the first argument u may be nonlinear while the text function ϕ is still linear.

Remark 5.4 (Semi-linear vs. semi-linear)**.** *The wording semi-linear should not be confused. A semi-linear form is a weak form of a nonlinear PDE in which the test function is still linear. Nonlinear equations can be distinguished into semi-linear, quasi-linear, fully nonlinear. Therefore, the notation of a semi-linear form does not indicate with which level of nonlinearity we are deadling with.*

5.5 Decoupled versus coupled

We shall specify how decoupled PDE systems can be characterized.

5.5.1 Definitions

Definition 5.5 (Coupled PDEs)**.** *A coupled PDE (system) consists of at least two PDEs that interact with each other. Namely, the solution of one PDE will influence the solution of the other PDE.*

Remark 5.6. *Coupled problems may also decouple by changing certain coefficients (e.g., Biot equations in porous media) or designing solution algorithms based on partitioned procedures. Due to decoupling, the single semi-linear forms may even become linear (e.g., Biot equations), but not always (e.g., fluid-structure interaction in which the subproblems NSE and elasticity are still nonlinear themselves).*

5.5.2 Coupling situations

The coupling can be of various types:

- Volume coupling (the PDEs live in the same domain) and exchange information via volume terms, right hand sides, coefficients.
  Example: Biot equations for modeling porous media flow, phase-field fracture, Cahn-Hilliard phase-field models for incompressible flows.
- Interface coupling (the PDEs live in different domains) and the information of the PDEs is only exchanged on the interface.
  Example: fluid-structure interaction.

The coupling is first of all based on first physical principles. Based on this information, appropriate numerical schemes can be derived. Specifically, interface-based coupling **is not simple!** Imagine a regular finite element mesh and assume that the interface cuts through the mesh elements.

In principle two approaches are distinguished:

- Interface-tracking: the interface is resolved in an exact fashion
- Interface-capturing: the interface moves through the mesh and detected with the help of an additional function (level-set or phase-field).

The situation becomes even more challenging when the coupled problems are modeled in different coordinate systems as it is the case for fluid-structure interaction. Fluids are described in Eulerian coordinates and solids usually in the Lagrangian framework. Here we first have to decide in which coordinate system we want to work. Four choices are possible:

- Eulerian (fluid) / Lagrangian (solid) - natural systems;

- Lagrangian/Eulerian (fluid) / Lagrangian (solid) - arbitrary Lagrangian-Eulerian;
- Lagrangian (fluid) / Eulerian (solid) - unnatural ! Not in use!
- Eulerian (fluid) / Eulerian (solid) - fully Eulerian.

5.6 Variational equations versus variational inequalities

Variational inequalities generalize weak forms such that constraints on the solution variables can be incorporated.

Let $(V, \|\cdot\|)$ be a real Hilbert space and K a nonempty, closed, convex subset of V. We define the mapping $A : K \to V^*$, where V^* is as usual the dual space of V. Then, we define the abstract problem:

Formulation 5.7 (Abstract variational inequality)**.** *Find $u \in K$ such that*

$$A(u)(\varphi - u) \geq 0 \quad \forall \varphi \in K.$$

If K is a linear subspace (and not only a set!) of V, then we have the linearity properties, see Section 3.12, which allow us to take

$$\varphi := u \pm w \in K$$

for each $w \in K$. Since the semi-linear form $A(\cdot)(\cdot)$ is linear in the second argument, we obtain:

$$A(u)(w) \geq 0 \quad \text{and} \quad A(u)(-w) \geq 0$$

from which follows:

$$A(u)(w) = 0 \quad \forall w \in K.$$

This shows that weak equations are contained in the more general setting of variational inequalities.

5.6.0.1 On the notation In the literature, the notation of the duality pairing is often found for describing weak forms and variational inequalities. This notation highlights better in which spaces we are working. Here, the previous statements read:

Formulation 5.8 (Abstract variational inequality - duality pairing notation)**.** *Find $u \in K$ such that*

$$\langle A(u), \varphi - u \rangle \geq 0 \quad \forall \varphi \in K.$$

Here we see more easily that $A(u) \in V^*$ and $\varphi - u \in V$.

If K is a linear subspace (and not only a set!) of V, then

$$\varphi := u \pm w \in K$$

for each $w \in K$. Since the semi-linear form $A(\cdot)(\cdot)$ is linear in the second argument, we obtain:

$$\langle A(u), w \rangle \geq 0 \quad \text{and} \quad \langle A(u), -w \rangle \geq 0$$

from which follows:

$$\langle A(u), w \rangle = 0 \quad \forall w \in K.$$

5.6.0.2 Minimization of functionals Let $\{V, \|\cdot\|\}$ be a reflexive, real Hilbert space and K be a nonempty closed convex subset of V and $F : K \to \mathbb{R}$ be a real functional defined on K.

Formulation 5.9 (Minimization problem)**.** *Find $u \in K$ such that*

$$F(u) = \inf_{v \in K} F(v)$$

or equivalently

$$F(u) \leq F(v) \quad \forall v \in K.$$

Definition 5.10 (Convexity)**.** *A functional $F : K \to \mathbb{R}$ is convex if and only if*

$$F(\theta u + (1-\theta)v) \leq \theta F(u) + (1-\theta)F(v) \quad u, v \in K, \quad 0 \leq \theta \leq 1.$$

The functional is strictly convex when $<$ holds true.

Definition 5.11. *Let*

$$\lim_{\varepsilon \to 0} \frac{d}{d\varepsilon} F(u + \varepsilon v) = F'(u)(v) = \langle F'(u), v \rangle$$

be the Gateaux derivative of F.

Theorem 5.12. *Let K be a subset of a normed linear space V and let $F : K \to \mathbb{R}$ be Gateaux differentiable. If u is a minimizer of F in K, then u can be characterized as follows:*

1. *If K is a nonempty closed convex subset of V, then*

$$F'(u)(v-u) \geq 0 \quad \forall v \in K.$$

2. *If K is a nonempty closed convex subset of V and $u \in int(K)$, then*

$$F'(u)(v) \geq 0 \quad \forall v \in K.$$

3. *If K is a linear variety with respect to w (i.e., K is a linear subspace translated by a factor w with respect to the origin), then*

$$F'(u)(v) = 0 \quad \forall v \in V$$

   and $v - w \in K$.

4. *If K is a linear subspace of V, then*

$$F'(u)(v) = 0 \quad v \in K.$$

Proof. Sketch.

1. The set K is convex. Thus $w := u + \theta(v-u) \in K$ for $\theta \in [0,1]$ and for every $u, v \in K$. If u is a minimizer of K, then we have

$$F(u) \leq F(w) = F(u + \theta(v-u)) \quad \forall v \in K.$$

   Differentiation yields:

$$\frac{d}{d\theta}[F(u + \theta(v-u)) - F(v)]|_{\theta=0} = F'(u)(v-u) \geq 0.$$

2. If $u \in int(K)$ and any $v \in K$. Then there exists $\varepsilon \geq 0$ such that $u + \varepsilon v \in K$. Therefore, we insert element in into the previous characterization and obtain:

$$F'(u)(u + \varepsilon v - u) = F'(u)(\varepsilon v) \geq 0.$$

   Since ε is arbitrary, this implies the assertion.

3. Let $K = \{w\} + M$, where M is a linear subspace of V. Then $w - v \in K$ implies $v \in M$. The same holds for $-v \in M$. Then,

$$F'(u)(v) = 0 \quad v \in V \quad \text{ such that } v - w \in K.$$

4. Take $w = 0$. Then,

$$F'(u)(v) = 0 \quad v \in K.$$

□

5.7 Coupled variational inequality systems (CVIS)

In multiphysics problems, several physical phenomena interact with each other, as for instance: solids, fluids, heat transfer, chemical reactions, and electromagnetic forces. The governing equations in these lecture notes are based on continuum mechanics. Such models often result in **nonstationary, nonlinear, coupled systems of partial differential equations (PDEs)** and they may be subject to inequality constraints. Using variational formulations, in [141], the resulting system is referred to as a **coupled variational inequality[3] system (CVIS)**. Characteristic features are:

1. At least two PDEs/VIs are involved;
2. Models in Lagrangian and Eulerian coordinates;
3. Different physical conservation laws such as balance of mass, momentum, angular momentum, and energy;
4. Second law of thermodynamics: entropy inequality principle characterizing the energy transfer direction in order to distinguish irreversible or reversible processes;
5. Information exchange between different PDEs/CVISs via:
   - coefficients,
   - linear or nonlinear coupling terms,
   - right-hand sides,
   - interfaces (internal boundaries).

Remark 5.13 (Coupled problems versus multiphysics)**.** *Phase-field fracture modeling as a key ingredient of multiphysics problems addressed in this book, consists of two interacting problems. However, phase-field fracture is not yet a multiphysics system because only one PDE describes physics, namely discplacements. The second problem is a VI, which is an indicator function that 'only' locates a discontinuity, damage or fracture. Therefore, phase-field fracture is simply a so-called coupled problem.*

5.8 Chapter summary and outlook

In the next section, we introduce finite differences as a first class of numerical methods for elliptic equations. In Section 8, we then discuss finite elements as another class of discretization methods.

[3] A variational inequality is referred to as VI.

6 Finite differences (FD) for elliptic problems

In this section, we address finite differences (FD) for boundary value problems (BVP). The key idea using FD is to approximate the derivatives of the PDEs with the help of difference quotients. Despite that finite differences are very old and other methods are better (in terms of domain shapes, general material coefficients) in state-of-the art approximations of PDEs, it is still worth to discuss them:

- They are very simple to understand and implement and yield a fast numerical solution and idea how a solution could look like;
- They are still used in (big) research codes in industry;
- Parts, such as difference quotients, serve as basis in all kind of software implementations of numerical methods for computing ODEs and PDEs in which derivatives need to be approximated.

First, we complete the missing properties of the continuous problem and show well-posedness for the one-dimensional Poisson problem with non-homogeneous Dirichlet boundary data. Then, we provide few words on the maximum principle and come afterwards to the discretization and finish by the numerical analysis.

6.1 A 1D model problem: Poisson

We study in this section a finite difference discretization for the one-dimensional Poisson problem:

$$\begin{cases} \text{Find} \quad u \in \mathcal{C}^2([0,L]) \quad \text{such that} \\ -u''(x) = f \quad ,\forall x \in (0,L), \\ u(0) = a, \\ u(L) = b, \end{cases} \tag{46}$$

where f is a given right hand side function $C^0([0,L])$ and $a, b \in R$.

For $f = -1$ and $a = 0$ and $L = 1$ we obtain a situation as sketched in Figure 12.

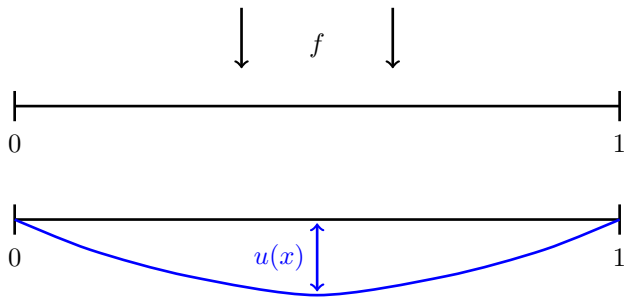


Figure 12: The clothesline problem: a uniform force $f = -1$ acts on a 1D line yielding a displacement $u(x)$.

Remark 6.1. *A derivation based on first principles in physics is provided in Section 9.6.2.*

6.2 Well-posedness of the continuous problem

We investigate in the following well-posedness, the maximum principle and the regularity of the continuous problem.

6.2.1 Green's function and well-posedness

We start with Green's function that represents the integral kernel of the Laplace operator (here the second-order derivative in 1D). Using Green's function, we obtain an explicit representation formula for the solution u. We define Green's function as:

$$G(x,s) = \begin{cases} \dfrac{s(L-x)}{L} & \text{if } 0 \le s \le x \le L, \\ \dfrac{x(L-s)}{L} & \text{if } 0 \le x \le s \le L. \end{cases} \tag{47}$$

We now integrate two times the differential equation (46). For all $\forall x \in [0, L]$, we obtain

$$u(x) = -\int_0^x \left(\int_0^t f(s)ds \right) dt + C_1 x + C_2, \tag{48}$$

where C_1 and C_2 are integration constants. Using Fubini's theorem (see e.g., [86]), we obtain

$$\int_0^x \left(\int_0^t f(s)ds \right) dt = \int_0^x \left(\int_s^x f(s)dt \right) ds = \int_0^x (x-s)\, f(s)ds$$

and therefore

$$u(x) = -\int_0^x (x-s)\, f(s)ds + C_1 x + C_2.$$

Using the boundary conditions from (46), we can determine C_1 and C_2:

$$\begin{aligned} u(x) &= -\int_0^x (x-s)f(s)ds + \frac{x}{L}\int_0^L (L-s)f(s)ds + a\frac{L-x}{L} + b\frac{x}{L} \\ &= \int_0^x \left(-\frac{L(x-s)}{L} + \frac{x}{L}(L-s) \right) f(s)ds + \frac{x}{L}\int_x^L (L-s)f(s)ds + a\frac{L-x}{L} + b\frac{x}{L} \\ &= \int_0^L G(x,s)f(s)ds \; + \; a\,\frac{(L-x)}{L} \; + b\,\frac{x}{L} \end{aligned} \tag{49}$$

in which we employed Green's function G defined in (47).

Now, we are able to investigate the well-posedness (see Section 2.8 for the meaning of well-posedness):

Proposition 6.2. *Problem* (46) *has a unique solution and depends continuously on the problem data via the following stability estimate:*

$$\|u\|_{C^2} = \max(\|u\|_{C^0}, L\|u'\|_{C^0}, L^2\|u''\|_{C^0}) \le C_1\|f\|_{C^0} + C_2|a| + C_3|b|,$$

where the constants may be different than before. If $a = b = 0$ (homogeneous Dirichlet conditions), the estimate simplifies to:

$$\|u\|_{C^2} \le C_1\|f\|_{C^0}.$$

An analoguous result for more general spaces is stated in Proposition 8.86.

Proof. Existence and uniquness follow immediately from the representation of the solution as

$$u(x) = \int_0^L G(x,s)f(s)\,ds \; + \; a\,\frac{(L-x)}{L} \; + b\,\frac{x}{L}. \tag{50}$$

The stability estimate follows via the following arguments. Green's function $G(x,s)$ is positive in $(0, L)$ and we can estimate as follows:

$$|u(x)| \le \|f\|_{C^0} \int_0^L G(x,s)\,ds + \; |a| \; + \; |b|$$

and $\forall x \in [0, L]$, we have

$$\int_0^L G(x,s)\,ds = \frac{(L-x)}{L}\int_0^x s ds + \frac{x}{L}\int_x^L (L-s)ds \tag{51}$$

$$\tag{52}$$

$$= \frac{(L-x)x^2}{2L} + \frac{x(L-x)^2}{2L} = \frac{(L-x)x}{2} \leq \frac{L^2}{8}, \quad \text{in } x = L/2, \tag{53}$$

yielding

$$\|u\|_{\mathcal{C}^0} = \max_{x\in[0,L]} |u(x)| \leq \frac{L^2}{8}\|f\|_{\mathcal{C}^0} + |a| + |b|.$$

All values on the right hand side are given by the problem data and are therefore known. However, the estimate is not yet optimal in the regularity of u. In the following we will see that we can even bound the second-order derivative of u by the right hand side f.

Indeed, by differentiating the solution (50) (actually better to see for the $G(x,s)$ function in (51)), we obtain $\forall x \in [0,L]$:

$$u'(x) = -\frac{1}{L}\int_0^x sf(s)ds + \frac{1}{L}\int_x^L (L-s)f(s)ds - \frac{a}{L} + \frac{b}{L}.$$

Since the integrand is still positive, we can again estimate:

$$|u'(x)| \leq \|f\|_{C^0}\frac{1}{L}\int_0^x s\,ds + \frac{1}{L}\int_x^L (L-s)ds + \frac{|a|}{L} + \frac{|b|}{L}$$

and $\forall x \in [0,L]$, the Green's function part yields:

$$\frac{1}{L}\int_0^x s\,ds + \frac{1}{L}\int_x^L (L-s)ds = \frac{x^2}{2L} + \frac{(L-x)^2}{2L} \leq \frac{L}{2}$$

thus

$$\|u'\|_{\mathcal{C}^0} = \max_{x\in[0,L]} |u'(x)| \leq \frac{L}{2}\|f\|_{\mathcal{C}^0} + \frac{|a|}{L} + \frac{|b|}{L}.$$

Differentiating a second time, we obtain (of course) $\forall x \in [0,L]$:

$$u''(x) = -f(x)$$

i.e., our original PDE. Therefore, we obtain the third estimate as:

$$\|u''\|_{\mathcal{C}^0} = \max_{x\in[0,L]} |u''(x)| \leq \|f\|_{\mathcal{C}^0}.$$

Consequently, the solution depends continuously on the problem data and therefore the problem is well-posed. □

Exercise 4. *Recapitulate and complete the above steps:*

- *Derive step-by-step the explicit form* (48)*;*
- *Determine* C_1 *and* C_2 *and convince yourself that* (49) *holds true;*
- *Convince yourself by an explicit computation that* $u''(x) = -f(x)$ *follows from* (50)*.*

6.2.2 Maximum principle on the continuous level

Since $G(x,s)$ is positive in $(0,L)$, the representation (50) yields immediately the maximum principle. That is to say:

$$f \geq 0,\, a \geq 0,\, b \geq 0 \quad \Rightarrow \quad u \geq 0.$$

Remark 6.3. *Intuitively this is immediately clear. Just consider a parabola function* $u(x) = \frac{1}{2}(-x^2 + x)$*, which is* $-u''(x) = 1$*, where* $f = 1$*. The function is positive with maximum in* 0.5 *and has its minima on the boundaries* 0 *and* 1*.*

Moreover, the maximum principle yields also uniqueness of the solution. Assume that u_1 and u_2 are two different solutions. Their difference is $w := u_1 - u_2$. Of course the original PDE, namely $-u''(x) = f$, also holds for w. Thus: $-w''(x) = f$. We apply f, a, b and $-f, -a, -b$ and obtain $-\Delta w = f$ from which we get $w \geq 0$ and $-\Delta(-w) = -f$ from which we get $w \leq 0$, respectively. Consequently, $w = 0$ and therefore $u_1 = u_2$.

6.2.3 Regularity of the solution

The regularity in the case of Poisson's problem carries over from the right hand side. If $f \in \mathcal{C}^0([0, L])$ we have shown that $u \in \mathcal{C}^2([0, L])$. For higher regularity $k \geq 1$, if $f \in \mathcal{C}^k([0, L])$, therefore for $1 \leq k' \leq k$, $u^{(k'+2)} = -f^{(k')} \in \mathcal{C}^0([0, L])$ and consequently $u \in \mathcal{C}^{k+2}([0, L])$. The solution is always two orders more regular than the right hand side data f.

6.3 Spatial discretization

After the previous theoretical investigations we address now a finite difference discretization for Poisson's problem. As previously mentioned, using finite differences, derivatives are approximated via difference quotients. For simplicity we consider homogeneous Dirichlet boundary values. We recall the corresponding problem:

Formulation 6.4.

$$\begin{cases} \textit{Find} \quad u \in \mathcal{C}^2([0, L]) \quad \textit{such that} \\ -u''(x) = f \quad \forall x \in (0, L), \\ u(0) = 0, \\ u(L) = 0. \end{cases} \tag{54}$$

In the first step, we need to discretize the domain by introducing discrete support points. Let $n \in \mathbb{N}$ and the step size (spatial discretization parameter) h (the distance between two support points) and the support points x_j be defined as

$$h = \frac{L}{n+1}, \quad x_j = jh \quad \text{for all } 0 \leq j \leq n+1.$$

By this construction, we have

- n inner points;
- $n + 2$ total points (including the two boundary points);
- $n + 1$ intervals (so-called mesh or grid elements).

The difference quotient for approximating a second order derivative is given by (recall classes to the introduction of numerical methods, e.g., [114]):

$$u''(x_j) \approx \frac{U_{j+1} - 2U_j + U_{j-1}}{h^2},$$

where we used a capitel letter U to distinguish the discrete solution from the exact solution u. In the following, be careful with the minus sign since we approximate the negative second order derivative of u:

$$-u''(x_j) \approx \frac{-U_{j+1} + 2U_j - U_{j-1}}{h^2}.$$

Consequently, the approximated, discrete, PDE reads:

$$\frac{-U_{j+1} + 2U_j - U_{j-1}}{h^2} = f(x_j), \quad 1 \leq j \leq n,$$

with $U_0 = U_{n+1} = 0$. Going from $j = 0, \dots n$, we obtain a linear equation system

$$\underbrace{\frac{1}{h^2}\begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix}}_{=A\in\mathbb{R}^{n\times n}} \underbrace{\begin{pmatrix} U_1 \\ U_2 \\ \vdots \\ \vdots \\ U_{n-1} \\ U_n \end{pmatrix}}_{=U\in R^n} = \underbrace{\begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ \vdots \\ f(x_{n-1}) \\ f(x_n) \end{pmatrix}}_{=F\in\mathbb{R}^n}$$

In compact form:

$$AU = F.$$

The 'only' remaining task consists now in solving this linear equation system. The solution U vector will then contain the discrete solution values on the corresponding support (inner) points.

Remark 6.5. *We could have formulated the above system for all points (including the two boundary points):*

$$\underbrace{\frac{1}{h^2}\begin{pmatrix} h^2 & 0 & & & \cdots & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & \cdots & 0 \\ \vdots & -1 & 2 & -1 & \ddots & & \vdots \\ & 0 & \ddots & \ddots & \ddots & & 0 \\ & \vdots & \ddots & -1 & 2 & -1 & 0 \\ & 0 & \cdots & 0 & -1 & 2 & -1 \\ 0 & 0 & \cdots & 0 & 0 & 0 & h^2 \end{pmatrix}}_{=A\in\mathbb{R}^{(n+2)\times(n+2)}} \underbrace{\begin{pmatrix} U_0 \\ U_1 \\ U_2 \\ \vdots \\ \vdots \\ U_{n-1} \\ U_n \\ U_{n+1} \end{pmatrix}}_{=U\in R^{n+2}} = \underbrace{\begin{pmatrix} 0 \\ f(x_1) \\ f(x_2) \\ \vdots \\ \vdots \\ f(x_{n-1}) \\ f(x_n) \\ 0 \end{pmatrix}}_{=F\in\mathbb{R}^{n+2}}$$

But we see immediately that we can save this slightly bigger matrix since the boundary values are already known and can be computed immediately.

6.4 Solving the linear equation system

Depending on the fineness of the discretization (the smaller h, the smaller the discretization error as we will see later), we deal with large matrices $A \in \mathbb{R}^{n\times n}$ for $n \sim 10^6$ and larger. For moderate sizes, a **direct** solution à la LU or Cholesky is competitive. For large n, due to

- computational cost,
- and memory restrictions,

we have to approximate U with the help of iterative solution schemes such as Jacobi, Gauss-Seidel, conjugate gradient, multigrid, and so forth. We come back to this topic in Section 10 in which we provide a basic introduction.

6.5 Well-posedness of the discrete problem

We now investigate well-posedness of the discrete problem.

6.5.1 Existence and uniqueness of the discrete problem

We first show that the matrix A is symmetric ($A = A^T$, where A^T is the transpose), positive definite. A direct calculation for two vectors $X, Y \in \mathbb{R}^n$ yields

$$\begin{aligned}
< AX, Y >_n &= \sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij} X_j Y_i \\
&= \frac{1}{h^2}\sum_{i=1}^{n}(-X_{i-1} + 2X_i - X_{i+1})Y_i \quad \text{(apply entries of } A \text{ to } X_j) \\
&= \frac{1}{h^2}\sum_{i=1}^{n}(X_i - X_{i+1})Y_i + \frac{1}{h^2}\sum_{i=1}^{n}(X_i - X_{i-1})Y_i \\
&= \frac{1}{h^2}\sum_{i=0}^{n}(X_i - X_{i+1})Y_i + \frac{1}{h^2}\sum_{i=1}^{n+1}(X_i - X_{i-1})Y_i \quad \text{(index shift due to bc)} \\
&= \frac{1}{h^2}\sum_{i=1}^{n+1}(X_{i-1} - X_i)Y_{i-1} + \frac{1}{h^2}\sum_{i=1}^{n+1}(X_i - X_{i-1})Y_i \\
&= \frac{1}{h^2}\sum_{i=1}^{n+1}(X_i - X_{i-1})(Y_i - Y_{i-1}) \\
&= \sum_{i=1}^{n+1}\frac{X_i - X_{i-1}}{h}\frac{Y_i - Y_{i-1}}{h} \\
&= \sum_{i=1}^{n+1}\frac{Y_i - Y_{i-1}}{h}\frac{X_i - X_{i-1}}{h} \quad \text{(switch factors)} \\
&= \ldots \quad \text{(do all steps backwards)} \\
&= < X, AY >_n,
\end{aligned}$$

with $X_0 = Y_0 = X_{n+1} = Y_{n+1} = 0$. This first calculation shows the symmetry. It remains to show that

$$< AX, Y >_n$$

is positive definite. We now consider the term

$$< AX, Y >= \ldots = \frac{1}{h^2}\sum_{i=1}^{n+1}(X_i - X_{i-1})(Y_i - Y_{i-1})$$

from the previous calculation. Setting $Y = X$ yields:

$$< AX, X >_n = \sum_{i=1}^{n+1}\left(\frac{X_i - X_{i-1}}{h}\right)^2 \geq 0.$$

This shows the positivity. It remains to show the definiteness: $< AX, X >_n = 0$ when $X = 0$. For all $1 \leq i \leq n+1$, we first obtain

$$X_0 = X_1 = \cdots = X_{i-1} = X_i = \cdots = X_n = X_{n+1},$$

and the zero boundary conditions finally yield $X = 0$. Consequently, the matrix A is symmetric positive definite. The eigenvalues are in $\mathbb{R}$ and strictly positive. Therefore, A is regular (recall the results from linear algebra!) and thus invertible, which finally yields:

Proposition 6.6 (Existence and uniqueness of the discrete problem)**.** *The discrete problem*

$$\frac{-U_{j+1} + 2U_j - U_{j-1}}{h^2} = f(x_j) \quad 1 \leq j \leq n, \tag{55}$$

with $U_0 = U_{n+1} = 0$, has one and only one solution for all $F \in \mathbb{R}^n$.

6.5.2 Maximum principle on the discrete level

As previously stated, the maximum principle on the continuous level has a discrete counterpart. Let F be given such that $F_j \geq 0$ for $1 \leq j \leq n$. We perform an indirecte proof and show that $\mu := \min_{1\leq j\leq n} U_j \geq 0$. We suppose $\mu < 0$ and take $i \in \{1, \cdots, n\}$ as one index (it exists at least one!) such that the minimum is taken: $U_i = \min_{1\leq j\leq n} U_j = \mu$. If $1 < i < n$, and U is solution of $AU = F$, the ith row can be written as:

$$2U_i - U_{i+1} - U_{i-1} = h^2 F_i \geq 0$$

under the assumption on F being positive. Therefore, we split

$$0 \leq (U_i - U_{i+1}) + (U_i - U_{i-1}) = (\mu - U_{i+1}) + (\mu - U_{i-1}) \leq 0$$

because $U_i = \mu$ is the minimum value, which we assumed to be negative. To satisfy this inequality chain it must hold:

$$U_{i-1} = U_i = U_{i+1} = \mu,$$

which shows that also the neighboring indices $i-1$ and $i+1$ satisfy the minimum. Extending this idea to all indices yields finally, $U_1 = U_n = \mu$. Therefore, we can investigate the first or the last row of the matrix A:

$$0 \leq F_1 = 2U_1 + U_2 = U_1 + (U_1 - U_2) \leq U_1 = \mu < 0.$$

But

$$0 \leq U_1 = \mu < 0$$

is a contradiction. Consequently, $\mu > 0$. Respecting the boundary conditions, we notice that for $U_0 = U_{n+1} = 0$ and $F_j \geq 0$ for all $1 \leq j \leq n$, we have the desired result

$$\mu = \min_{0\leq j\leq n+1} U_j \geq 0$$

showing that we have a discrete maximum principle.

Proposition 6.7 (Monotone inverse)**.** *The matrix A corresponding to the finite difference discretization is a so-called M matrix (inverse-monotone), i.e., the inverse A^{-1} is element-wise non-negative, i.e.,*

$$A^{-1} \geq 0.$$

Proof. For $1 \leq j \leq n$, we fix $F^j = (0 \cdots 1 \cdots 0) = (\delta_{ij})_{1\leq i\leq n}$, where δ_{ij} is the Kronecker symbol and X^j is the solution of $AX^j = F^j$. By construction, $X^j = A^{-1}F^j$ and for $1 \leq i \leq n$, the ith row can be written as

$$X_i^j = \sum_{k=1}^{n} A_{ik}^{-1} F_k^j = \sum_{k=1}^{n} A_{ik}^{-1}\delta_{kj} = A_{ij}^{-1}. \tag{56}$$

Using the maximum principle, we know that for $1 \leq i, j \leq n$ and $F_i^j = \delta_{ij} \geq 0$, the solution is positive, i.e.,

$$X_i^j \geq 0$$

(j is the jth solution and i is the component of that solution). Thanks to (56), we have $X_i^j = A_{ij}^{-1}$, and we obtain

$$A_{ij}^{-1} \geq 0,$$

which means that the inverse of the matrix A is an M matrix. □

6.6 Numerical analysis: consistency, stability, and convergence

We finally investigate the convergence properties of the finite difference discretization. As we know from ODE numerics, for linear schemes, consistency plus stability will yield convergence.

6.6.1 Basics in error estimation

We recall the basics. Let $\bar{A}\bar{x} = \bar{b}$ the continuous problem and $Ax = b$ the discrete approximation. Then, we define:

- Approximation error: $e := \bar{x} - x$
- Truncation error: $\eta := A\bar{x} - b$ (exact solution into numerical scheme)
- Residual (defect): $\rho := \bar{A}x - \bar{b}$ (discrete solution into continuous problem)

Furthermore, when the underlying numerical scheme is linear (see ODE lectures), we have:

- Truncation error plus stability yields convergence.

In the following, it will be the same procedure that we use.

6.6.2 Truncation error

We first derive an expression for the **truncation error** η (the local discretization error), which is obtained by plugging in the exact (unknown) solution into the numerical scheme.

Let

$$e_j = U_j - \bar{U}_j \quad \text{for } 1 \leq j \leq n,$$

where $\bar{U}_j = (\bar{U}_i)_{1\leq i\leq n} = (u(x_i))_{1\leq i\leq n} \in \mathbb{R}^n$. Then, we have for $1 \leq i \leq n$:

$$\begin{aligned}(Ae)_i &= \left[A(U-\bar{U})\right]_i = (F - A\bar{U})_i = f(x_i) - (A\bar{U})_i \\ &= -u''(x_i) - \frac{-\bar{U}_{i+1} + 2\bar{U}_i - \bar{U}_{i-1}}{h^2} \\ &= \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} - u''(x_i)\end{aligned}$$

and thus

$$Ae = \eta \tag{57}$$

where $\eta \in \mathbb{R}^n$ is the local truncation error that arises because the exact solution u does not satisfy the numerical scheme (it is only the discrete solution U that is a solution of the discrete scheme!). Consequently, for all $1 \leq i \leq n$, we have

$$\eta_i := \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} - u''(x_i).$$

6.6.3 Consistency

The consistency is based on estimates of the local truncation error. The usual method to work with is a Taylor expansion (see Section 3.14). We have previously established that for $f \in \mathcal{C}^2([0, L])$, we have $u \in \mathcal{C}^4([0, L])$. Consequently, we can employ a Taylor expansion up to order 4 at the point $x_i, 1 \leq i \leq n$. This brings us to:

$$u(x_{i+1}) = u(x_i) + hu'(x_i) + \frac{h^2}{2}u''(x_i) + \frac{h^3}{6}u^{(3)}(x_i) + \frac{h^4}{24}u^{(4)}(\tau_i^+)$$

$$u(x_{i-1}) = u(x_i) - hu'(x_i) + \frac{h^2}{2}u''(x_i) - \frac{h^3}{6}u^{(3)}(x_i) + \frac{h^4}{24}u^{(4)}(\tau_i^-)$$

with $\tau_i^+ \in [x_i, x_{i+1}]$ and $\tau_i^- \in [x_{i-1}, x_i]$. It follows that for all $1 \leq i \leq n$

$$|\eta_i| = \left| \frac{h^2}{24} u^{(4)}(\tau_i^+) + \frac{h^2}{24} u^{(4)}(\tau_i^-) \right| \leq \frac{h^2}{12} \|u^{(4)}\|_{C^0} = \frac{h^2}{12} \|f''\|_{C^0}$$

in which we used that $u^{(4)} = -f''$. As final result, we obtain:

Proposition 6.8. *The local truncation error of the finite difference approximation of Poisson's problem can be estimated as*

$$\|\eta\|_\infty := \max_{1 \leq i \leq n} |\eta_i| \leq C_f \, h^2, \quad \text{with } C_f = \frac{1}{12} \|f''\|_{C^0} \tag{58}$$

Therefore, the scheme (55) *is consistent with order* 2.

Remark 6.9. *We notice that the additional order of the scheme (namely order two and not one) has been obtained since we work with a central difference quotient to approximate the second order derivative. Of course, here it was necessary to assume sufficient regularity of the right hand side f and the solution u.*

Remark 6.10. *The requirements on the regularity of the solution u to show the above estimates is much higher than the corresponding estimates for finite elements (see Section 8.13). In practice, such a high regularity can only be ensured in a few cases, which is a drawback of finite differences in comparison to finite elements. But still, we can use the method in practice, but one has to be careful whether the results are still robust and reliable.*

6.6.4 Stability in L^∞

We now investigate the stability of the finite difference scheme. We recapitulate the matrix norm $\|.\|_\infty$ (maximum row sum) for $M \in \mathbb{R}^{n,n}$ defined by

$$\|M\|_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^{n} |M_{ij}|$$

Moreover, we denote by $w(x)$ the exact solution for the problem in which $f = 1$ on $[0, L]$. The choice of $f = 1$ is only for simplicity. More general f work as well, but will require more work in the proof below.

Proposition 6.11. *It holds:*

$$\|A^{-1}\|_\infty = \|\bar{W}\|_\infty,$$

and

$$\|A^{-1}\|_\infty \leq \frac{L^2}{8}.$$

Proof. For $f = 1$, the second derivative vanishes: $f'' = 0$. From Section 6.6.3 with (58) we know that in this case

$$\|\eta\|_\infty = 0$$

and therefore

$$0 = Ae = A(W - \bar{W})$$

where $W \in \mathbb{R}^n$ denotes the discrete solution corresponding to $f = 1$ (the full right hand side vector is denoted by F_1 in the following). Moreover, we denote $\bar{W} = (w(x_i))_{1 \leq i \leq n} \in \mathbb{R}^n$ the continuous solution. Consequently,

$$\begin{aligned} A(W - \bar{W}) = 0 \quad &\Rightarrow \quad AW = A\bar{W} \quad \Rightarrow \quad F_1 = A\bar{W} \\ &\Rightarrow \quad \bar{W} = A^{-1}F_1 = A^{-1}. \end{aligned}$$

The last step holds because we work with $f = 1$ and therefore $F_1 = 1$. For $1 \leq i \leq n$, we obtain

$$\bar{W}_i = \sum_{j=1}^{n} A_{ij}^{-1} (F_1)_i = \sum_{j=1}^{n} A_{ij}^{-1} = \sum_{j=1}^{n} |A_{ij}^{-1}|$$

The last step holds because A is an M-matrix. Otherwise, we would not be allowed to take absolute signs. We finally obtain:

$$\|A^{-1}\|_\infty = \max_{1\leq i\leq n} \sum_{j=1}^{n} |A^{-1}_{ij}| = \max_{1\leq i\leq n} |\bar{W}_i| = \|\bar{W}\|_\infty.$$

The exact solution to the problem $-w''(x) = 1$ on $(0, L)$ with homogeneous Dirichlet conditions is $w(x) = \dfrac{x(L-x)}{2}$, which atteins its maximum at $x = L/2$. This yields $\max(w) = L^2/8$ and therefore,

$$\|A^{-1}\|_\infty \leq \frac{L^2}{8},$$

which shows the assertion. □

Remark 6.12. *The case $f = 1$ with $\eta = Ae = 0$ has a very practical implication. Here, the discrete solution matches the continuous solution in the support points $x_i, 1 \leq i \leq n$. This can be confirmed by numerical simulations; see e.g., Setion 6.7. But be careful, this correspondance is in general only true for $f'' = 0$ and only in 1D cases.*

Remark 6.13. *The previous result can be extended to other right hand sides f, which are not necessarily $f = 1$.*

6.6.5 Convergence L^∞

With the previous two results, we can now proof convergence of the finite difference scheme. We assume as before $f \in \mathcal{C}^2([0, L])$ and obtain:

Proposition 6.14. *For the convergence of the finite difference scheme* (55)*, it holds:*

$$\|e\|_\infty = \|A^{-1}\eta\|_\infty \leq \|A^{-1}\|_\infty \|\eta\|_\infty \leq \frac{L^2}{8} C_f\, h^2 = \frac{L^2}{96}\|f''\|_{\mathcal{C}^0} h^2 = O(h^2),$$

where we employed the stability and consistency estimates and (58)*. Consequently, we have quadratic convergence as h tends to zero.*

6.6.6 Convergence L^2

In this final section, we proof a second convergence result, but now in the L^2 norm. We define two norms:

$$\|X\|_A = \left(\sum_{j=1}^{n+1} \frac{1}{h}|X_j - X_{j-1}|^2\right)^{\frac{1}{2}} \qquad \|X\|_2 = \left(\sum_{j=1}^{n} h|X_j|^2\right)^{\frac{1}{2}}$$

and notice that $< . >_A$ and $< . >_2$ are the corresponding scalar products.

Proposition 6.15. *It holds for each $X \in R^n$:*

$$\|X\|_2 \leq L^{1/2}\|X\|_\infty \leq L\|X\|_A.$$

Proof. Let $X \in \mathbb{R}^n$ for all $1 \leq i \leq n$ and we set as usually $X_0 = X_{n+1} = 0$. Then:

$$\begin{aligned} |X_i| &= \left|X_0 + \sum_{j=1}^{i}(X_j - X_{j-1})\right| = \sum_{j=1}^{i} h\frac{X_j - X_{j-1}}{h} * 1 =< \frac{X_j - X_{j-1}}{h}, 1 >_i \\ &\leq \left(\sum_{j=1}^{i} h\left|\frac{X_j - X_{j-1}}{h}\right|^2\right)^{1/2}\left(\sum_{j=1}^{i} h|1|^2\right)^{1/2} \\ &\leq \|X\|_A L^{1/2} \end{aligned}$$

where we used the Cauchy-Schwarz inequality. It follows that

$$\|X\|_\infty \leq L^{1/2}\|X\|_A.$$

Moreover,

$$\|X\|_2^2 = \left(\sum_{j=1}^{n} h|X_j|^2\right) \leq \|X\|_\infty^2 \left(\sum_{j=1}^{n} h\right) \leq \|X\|_\infty^2 L.$$

We finally obtain

$$\|X\|_2 \leq \|X\|_\infty L^{1/2} \leq L\|X\|_A.$$

□

Remark 6.16. *We notice that* $\|X\|_2 \leq L\|X\|_A$ *holds because* $X_0 = 0$ *and* $\sum_{j=1}^{i} h|1|^2$ *is bounded. Otherwise the previous result would be not correct.*

Proposition 6.17. *Let* $X, Y \in \mathbb{R}^n$*. For a symmetric, positive, definite matrix* A*, we can define the so-called* A*-norm; see also Section 10.4. It holds*

$$< X, Y >_A = < AX, Y >_2$$

and

$$\|e\|_A \leq L\|\eta\|_2.$$

Proof. We already have shown that for all $X, Y \in \mathbb{R}^n$, it holds

$$< AX, Y >_n = \sum_{i=1}^{n+1} \frac{X_i - X_{i-1}}{h} \frac{Y_i - Y_{i-1}}{h}.$$

Therefore,

$$< AX, Y >_2 = h < AX, Y >_n = h \sum_{i=1}^{n+1} \frac{X_i - X_{i-1}}{h} \frac{Y_i - Y_{i-1}}{h} = < X, Y >_A$$

from which follows

$$\|e\|_A^2 = < e, e >_A = < Ae, e >_2 = < \eta, e >_2 \leq \|\eta\|_2\|e\|_2 \leq L\|\eta\|_2\|e\|_A$$

in which we used again the Cauchy-Schwarz inequality. This shows the assertion. □

Proposition 6.18. *It holds:*

$$\|e\|_A \leq L\|\eta\|_2 \leq L^{3/2}\|\eta\|_\infty \leq L^{3/2} C_f\, h^2$$

and

$$\|e\|_2 \leq L\|e\|_A \leq L^{5/2} C_f\, h^2.$$

In other words: the scheme (55) *converges in the* L^2 *norm and the* A *norm with order* 2 *when* h *tends to zero.*

6.7 Numerical test: 1D Poisson

We finish this 1D section with a numerical test implemented in octave [44].

6.7.1 Homogeneous Dirichlet boundary conditions

Formulation 6.19. *Let* $\Omega = (0, 1)$

$$\begin{cases} \text{Find} \quad u \in \mathcal{C}^2(\Omega) \quad \text{such that} \\ -u''(x) = f \quad \text{in } \Omega \\ u(0) = 0, \\ u(1) = 0. \end{cases} \tag{59}$$

where $f = -1$. *The continuous solution can be computed here (see Section 8.1.1) and is*

$$u(x) = \frac{1}{2}(-x^2 + x).$$

For the discretization we choose $n = 4$. *Thus we have five support points*

$$x_0, x_1, x_2, x_3, x_4$$

and $h = 1/4 = 0.25$.

With these settings we obtain for the matrix A (see Section (6.3)) including boundary conditions:

```
1     0    0    0    0
-16  32  -16    0    0
0   -16   32  -16    0
0     0  -16   32  -16
0     0    0    0    1
```

and for the right hand side vector b:

```
0
-1
-1
-1
0
```

We use the famous backslash solution operator (in fact an LU decomposition):

```
U = A\b
```

and obtain as solution $U = (U_0, U_1, U_2, U_3, U_4)$:

```
 0.00000
-0.09375
-0.12500
-0.09375
 0.00000
```

A plot of the discrete solution is provided in Figure 13. Moreover, the continuous solution can be evaluated in the support as well. Here, we see that both solutions match. This was already suggested by our numerical analysis; see in particular Remark 6.12.

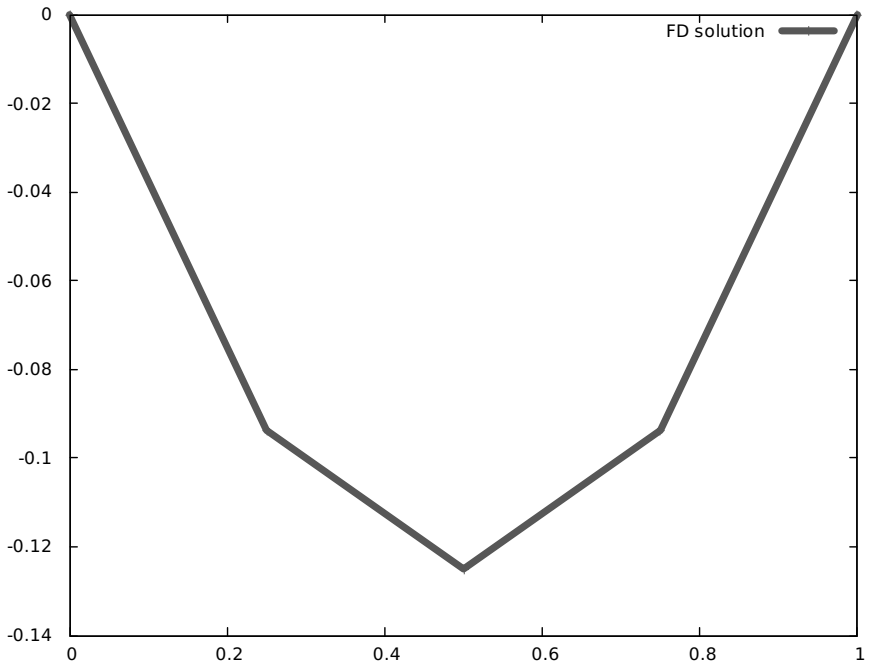


Figure 13: Solution of the 1D Poisson problem with $f = -1$ using five support points with $h = 0.25$. We observe that the approximation is rather coarse. A smaller h would yield more support points and more solution values and therefore a more accurate solution.

Remark 6.20 (Matrix A in FD vs. FEM)**.** *In the above matrix A and RHS b, we have implicitly $1/h^2$ and 1 as factors, respectively. Indeed*

$$A_{FD} = \frac{1}{h^2}\begin{pmatrix} h^2 & 0 & & & \cdots & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & \cdots & 0 \\ \vdots & -1 & 2 & -1 & \ddots & & \vdots \\ & 0 & \ddots & \ddots & \ddots & & 0 \\ & \vdots & \ddots & -1 & 2 & -1 & 0 \\ & 0 & \cdots & 0 & -1 & 2 & -1 \\ 0 & 0 & \cdots & 0 & 0 & 0 & h^2 \end{pmatrix}, \qquad b_{FD} = (0, 1, \ldots, 1, 0)^T, \qquad \frac{1}{h^2} = \frac{1}{16}.$$

The corresponding matrix using an FEM scheme (Section 8) looks slightly different, but is in fact the same. Here, we integrate only u', but multiplied with a test function φ'. On the right hand side we also have φ. Here, we obtain:

$$A_{FEM} = \frac{1}{h}\begin{pmatrix} h & 0 & & & \cdots & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & \cdots & 0 \\ \vdots & -1 & 2 & -1 & \ddots & & \vdots \\ & 0 & \ddots & \ddots & \ddots & & 0 \\ & \vdots & \ddots & -1 & 2 & -1 & 0 \\ & 0 & \cdots & 0 & -1 & 2 & -1 \\ 0 & 0 & \cdots & 0 & 0 & 0 & h \end{pmatrix}, \qquad b_{FEM} = h * (0, 1, \ldots, 1, 0)^T, \qquad \frac{1}{h} = \frac{1}{4}.$$

Dividing over h in A_{FEM} and b_{FEM} yields directly A_{FD} and b_{FD}.

6.7.2 Nonhomogeneous Dirichlet boundary conditions $u(0) = u(1) = 1$

We change the boundary conditions to

$$u(0) = u(1) = 1.$$

The right hand side then reads $b = (1, -1, -1, -1, 1)^T$. We obtain as solution:

```
1.00000
0.90625
0.87500
0.90625
1.00000
```

6.7.3 Nonhomogeneous Dirichlet boundary conditions $u(0) = 3$ and $u(1) = 2$

We change the boundary conditions to

$$u(0) = 3 \quad , u(1) = 2.$$

The right hand side then reads $b = (3, -1, -1, -1, 2)^T$.

We obtain as solution:

```
3.0000
2.6562
2.3750
2.1562
2.0000
```

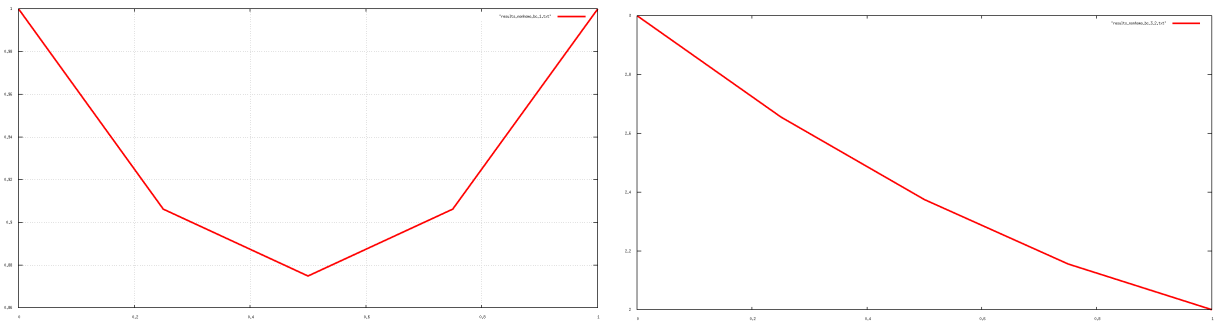


Figure 14: Solution of the 1D Poisson problem with $f = -1$ and nonhomogeneous boundary conditions.

6.7.4 How can we improve the numerical solution?

An important question is how we can improve the numerical solution? A five-point approximation as employed in Figure 13 is obviously not very accurate. After all previous subsections in this chapter, the answer is clear: increasing the number of support points x_n, i.e., $n \to \infty$ in the given domain Ω will yield $h \to 0$ (recall $h = x_{j+1} - x_j$). According to our findings in the numerical analysis section, the discretization error will become very small and we have therefore an accurate approximation of the given problem. On the other hand, large n will require a lot of computational resources. We have finally to find a compromise between accuracy of a numerical solution and computational cost (recall the concepts outlined in Section 2.6).

6.8 Finite differences in 2D

We extend the idea of finite differences now to two-dimensional problems. The goal is the construction of the method. For a complete numerical analysis we refer to the literature, e.g., [65, 106].

In this section we follow (more or less) [111]. We consider

$$-\Delta u = f \quad \text{in } (0,1)^2, \tag{60}$$
$$u = u_D \quad \text{on } \partial\Omega_D. \tag{61}$$

6.8.1 Discretization

In extension to the previous section, we now construct a grid in two dimensions. The local grid size is h. We define

$$\Omega_h := \{(x_{ij}), i,j = 0,\dots,N |\ x_{ij} = (ih, jh), h = N^{-1}\}$$

We seek a solution in all inner grid points

$$\Omega_h^0 := \{(x_{ij}), i,j = 1,\dots,N-1 |\ x_{ij} = (ih, jh), h = N^{-1}\}$$

For the following, we introduce a grid function $u_h := (u_{ij})_{i,j=1}^{N-1}$ and seek unknown function as the solution of

$$-\Delta_h u_h = F_h, \quad x_{ij} \in \Omega_h^0.$$

Using a central finite difference operator of second order we obtain

$$-(\Delta_h u_h)_{ij} = \frac{1}{h^2}(4u_{ij} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1}), \quad (F_h)_{ij} := f(x_{ij})$$

On the boundary, we set

$$u_{0,j} = u_{N,j} = u_{i,0} = u_{i,N}$$

This difference quotient is known as a five-point stencil:

$$S_h = \frac{1}{h^2}\begin{pmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{pmatrix}$$

In total we obtain the following linear equation system:

$$A_h u_h = f_h \tag{62}$$

with

$$A_h := \begin{pmatrix} B_m & -I_m & & \\ -I_m & B_m & -I_m & \\ & -I_m & B_m & \ddots \\ & & \ddots & \end{pmatrix}, \quad B := \begin{pmatrix} 4 & -1 & & \\ -1 & 4-1 & & \\ & -1 & 4 & \ddots \\ & & \ddots & \end{pmatrix} \tag{63}$$

The matrix has size $A_h \in \mathbb{R}^{N^2 \times N^2}$ and bandwidth $2N+1$. Further properties are:

Proposition 6.21. *The matrix A_h has the following properties:*

1. *A_h has five entries per row;*
2. *it is symmetric;*
3. *weakly diagonal-dominant;*
4. *positive definite;*
5. *an M matrix.*

It holds:

Proposition 6.22. *The five-point stencil approximation of the Poisson problem in two dimensions satisfies the following a priori error estimate:*

$$\max_{ij} |u(x_{ij}) - u_{ij}| \leq \frac{1}{96} h^2 \max_{\Omega} (|\partial_x^4 u| + |\partial_y^4 u|).$$

Proof. Later. □

The resulting linear equation system $A_h u_h = F_h$ is sparse in the matrix A_h, but maybe large, for large N (i.e., very small h and consequently high accuracy).

To determine the condition number (which is important for the numerical solution), we recall the eigenvalues and eigenvectors:

$$\lambda_{kl} = h^{-2}(4 - 2(cos(kh\pi) + cos(lh\pi))),$$
$$\omega_{kl} = (\sin(ikh\pi)\sin(jlh\pi))_{ij=1}^{N-1}.$$

The largest and smallest eigenvalues are, respectively:

$$\lambda_{max} = h^{-2}(4 - 4\cos(1-h)\pi) = 8h^{-2} + O(1),$$
$$\lambda_{min} = h^{-2}(4 - 4\cos(h\pi)) = h^{-2}(4 - 4(1 - 0.5(\pi^2 h^2)) + O(h^2) = 2\pi^2 + O(h^2).$$

Therefore, the condition number is

$$cond_2(A_h) = \frac{\lambda_{max}}{\lambda_{min}} = \frac{2}{\pi^2 h^2},$$

which is similar to the heat equation.

Remark 6.23 (on the condition number)**.** *The order $O(h^{-2})$ of the matrix A_h is determined by the differential equation and **neither** by the dimension of the problem (here 2) **nor** the quadratic order of the consistency error. This statement holds true for finite differences as well as finite elements; see also Section 10.1.*

6.9 Finite differences for time-dependent problems: 1D space / 1D time

We extend our developments to time-dependent problems and restrict ourselves again to one-dimensional spatial problems. As model problem we consider the heat equation. As spatial interval, we choose $\Omega = (0, 1)$. The time interval is denoted by $I := (0, T)$ where $T > 0$ is the end time value. The problem reads:

Formulation 6.24 (Heat equation)**.** *Let $\nu > 0$ be a material parameter. Find $u := u(x, t)$ such that*

$$\partial_t u - \nu \partial_{xx} u = 0 \quad \text{in } \Omega \times I \qquad \text{PDE}, \tag{64}$$
$$u(t, 0) = u(t, 1) \quad \text{on } \partial\Omega \qquad \text{Boundary condition}, \tag{65}$$
$$u(0, x) = u_0(x) \quad \text{in } \Omega \times \{0\} \qquad \text{Initial condition}. \tag{66}$$

The PDE is complemented with a boundary condition of Dirichlet type, here a 1-periodic boundary condition, and an intial condition for time $t = 0$.

6.9.1 Algorithmic aspects: temporal and spatial discretization

For spatial and temporal discretization, we introduce a spatial mesh parameter h and a temporal mesh parameter k:

$$h = \frac{1}{(N+1)} > 0, \qquad k > 0.$$

We define the (mesh) nodes

$$(t_n, x_j) = (nk, jh) \quad \forall n \geq 0,\ j \in \{0, 1, \ldots, N+1\}.$$

A discrete value that approximates the (unknown) exact solution in a point (t_n, x_j) is denoted by u_j^n. The initial value is discretized by

$$u_j^0 = u_0(x_j), \quad j \in \{0, 1, \ldots, N+1\}.$$

As boundary values we have in the discrete sense

$$u_0^n = u_{N+1}^n \quad \forall n > 0.$$

To this end, at each time point, we compute the values

$$(u_j^n)_{1 \le j \le N} \in \mathbb{R}^N$$

In the following we introduce some schemes (i.e., algorithms) in order to compute these solution values. Each scheme yields a system of N equations (for each spatial point x_j for $1 \le j \le N$) in order to compute the N components of u_j^n.

Definition 6.25 (Explicit scheme)**.** *An explicit scheme (explicit in terms of n) is*

$$\frac{u_j^{n+1} - u_j^n}{k} + \nu \frac{-u_{j-1}^n + 2u_j^n - u_{j+1}^n}{h^2} = 0$$

for $n > 0$ and $j \in \{1, \dots, N\}$. This scheme is known from ordinary differential equations as the forward Euler scheme .

Definition 6.26 (Implicit scheme)**.** *An implicit scheme (implicit in terms of n, i.e., the solution is not immediately obtained) is*

$$\frac{u_j^{n+1} - u_j^n}{k} + \nu \frac{-u_{j-1}^{n+1} + 2u_j^{n+1} - u_{j+1}^{n+1}}{h^2} = 0$$

for $n > 0$ and $j \in \{1, \dots, N\}$. This scheme is the well-known backward Euler scheme.

It must be further justified that indeed the implicit scheme can be used the values u_j^{n+1}. To this end, we have to write out explicitly all N equations, yielding a linear equation system. Now, we need to check whether the resulting matrix is invertible. Specifically, we have

$$\begin{pmatrix} 1+2c & -c & & & 0 \\ -c & 1+2c & -c & & \\ & \ddots & \ddots & \ddots & \\ & & -c & 1+2c & -c \\ 0 & & & -c & 1+2c \end{pmatrix}$$

with $c := \nu \frac{k}{h^2}$.

A combination of the two previous schemes yields a famous scheme, the so-called θ scheme:

Definition 6.27 (One-Step-θ scheme)**.** *The θ scheme is defined as:*

$$\frac{u_j^{n+1} - u_j^n}{k} + \nu\theta \frac{-u_{j-1}^{n+1} + 2u_j^{n+1} - u_{j+1}^{n+1}}{h^2} + \nu(1-\theta) \frac{-u_{j-1}^n + 2u_j^n - u_{j+1}^n}{h^2} = 0$$

for $n > 0$ and $j \in \{1, \dots, N\}$. Here the classification (whether implicit or explicit) depends on the choice of $\theta \in [0, 1]$. For $\theta = 0$ the scheme is clearly explicit. For $\theta > 0$ the scheme is implicit, but care has to be taken about the meaning. Indeed for $0 \le \theta \le 0.5$, it can be shown that the scheme has problems in robustness. For $\theta \ge 0.5$, we have a better properties. Indeed, for $\theta = 1$, we obtain the very stable (implicit) backward Euler scheme. Finally for $\theta = 0.5$, the resulting scheme has similarities with the trapezoidal rule in numerical quadrature, can specifically in application of the heat equation it is known as Crank-Nicolson scheme..

Remark 6.28. *Further schemes can be obtained with the same rules known from ordinary differential equations.*

6.9.2 Consistency and precision

As before, based on Taylor developments, we now investigate the consistency of our numerical schemes and also their precision. In abstract form we write the PDE as (see also Definition 4.46):

$$F(u) = 0$$

In a general fashion, the corresponding FD scheme can be written as:

$$F_{k,h}(u_{j+i}^{n+m}) = 0 \quad \text{for } m^- \leq m \leq m^+, i^- \leq i \leq i^+$$

The values m^-, m^+, i^-, i^+ define the size of the so-called **stencil**

Definition 6.29 (Truncation error)**.** *The finite difference scheme $F_{k,h}(u_{j+i}^{n+m}) = 0$ is consistent with the PDE $F(u) = 0$ when for all sufficiently smooth solutions $u(t,x)$, the truncation error, it holds*

$$F_{k,h}(u(t+mk, x+ih)) \to 0 \quad \textit{for } h \to 0, k \to 0.$$

Furthermore, the order is

$$F_{k,h}(u(t+mk, x+ih)) = O(h^p + k^q)$$

We say then that the finite difference scheme is consistent with order p in space and order q in time when the last approximation holds true.

Remark 6.30. *The truncation error is always obtained by inserting the exact solution into the numerical scheme.*

Proposition 6.31. *For the forward Euler scheme, it holds $q = 1$ (first order in time) and $p = 2$ (second order in space).*

Proof. We choose a function $u \in C^6$ and perform Taylor. Then, we re-organize all terms such that we obtain the original PDE. The remainder terms constitute the order of the scheme. □

6.9.3 Stability

We now concentrate on the stability. As outlined in Section 2.6, the non-stability often results in non-physical oscillations of the numerical solution. The investigation of stability in PDEs for finite differences follows simply the concepts of those learned in ODEs. For the stability, we need to define proper norms to work with. We recall first that $u^n = (u_j^n)_{1 \leq j \leq N}$ is our numerical solution. We define the following norm:

$$\|u^n\|_p = \Big(\sum_{j=1}^{N} h|u_j^n|^p\Big)^{1/p}$$

for $1 \leq p \leq \infty$. For $p = \infty$, we use

$$\|u^n\|_\infty = \max_{1 \leq j \leq N} |u_j^n|$$

Due to the weighting with h, the norm $\|u^n\|_p$ is identical with the norm $L^p(0,1)$ for constant function on the half-open sub-interval $[x_j, x_{j+1})$.

Definition 6.32 (Stability)**.** *A finite difference scheme is **(unconditionlly) stable** with respect to the norm $\|\cdot\|$ defined before, if there exists a constant $K > 0$ independent of the mesh sizes k and h such that*

$$\|u^n\| \leq K\|u^0\| \quad \textit{for } n \geq 0.$$

*Here u^n is the numerical solution at time step t^n and u^0 the initial value. A finite difference scheme is **conditionally stable** if the stability estimate only holds under certain conditions on h and k.*

Definition 6.33 (Linear numerical schemes)**.** *A finite difference scheme is **linear** when $F_{k,h}(u_{j+i}^{n+m})$ is linear with respect to its argument u_{j+i}^{n+m}.*

For linear one-step schemes, the stability can be investigated in the equivalent formulation

$$u^{n+1} = Au^n$$

where A is a linear operator (a matrix) from $\mathbb{R}^N$ mapping to $\mathbb{R}^N$. For instance, we know already for the forward Euler scheme:

$$B := \begin{pmatrix} 1+2c & -c & & & 0 \\ -c & 1+2c & -c & & \\ & \ddots & \ddots & \ddots & \\ & & -c & 1+2c & -c \\ 0 & & & -c & 1+2c \end{pmatrix}$$

The matrix A is nothing else than the inverse of B:

$$A = \begin{pmatrix} 1-2c & c & & & 0 \\ c & 1-2c & c & & \\ & \ddots & \ddots & \ddots & \\ & & c & 1-2c & c \\ 0 & & & c & 1-2c \end{pmatrix}$$

If we now iterate, we obtain

$$u^n = A^n u^0$$

Consequently, the stability becomes

$$\|A^n u^0\| \leq K\|u^0\| \quad \text{for } n \geq 0,\ \forall u^0 \in \mathbb{R}^N.$$

From introductory classes into numerical analysis, we know the definition of matrix norm:

$$\|M\| = \sup_{u \in R^N, u \neq 0} \frac{\|Mu\|}{\|u\|}$$

Hence, the stability of the scheme is equivalent to

$$\|A^n\| \leq K, \quad n \geq 0,$$

which means nothing else than that the powers of A are bounded.

6.9.4 Stability in L^∞

In the following, we investigate the stability in the L^∞. This is linked to the discrete maximum principle.

Definition 6.34. *A finite difference scheme satisfies the discrete maximum principle if for all $n \geq 0$ and all $1 \leq j \leq N$, it holds*

$$\min\left(0, \min_{0 \leq j \leq N+1} u_j^0\right) \leq u_j^n \leq \max\left(0, \max_{0 \leq j \leq N+1} u_j^0\right)$$

Using the maximum principle, we obtain the following result:

Proposition 6.35. *For forward Euler scheme is stable in the L^∞ norm if and only if the CFL (Courant-Friedrichs-Lewy) condition*

$$2\nu k \leq h^2$$

is satisfied. The implicit Euler scheme is unconditionally stable in the L^∞ norm.

6.9.5 Convergence

Having the consistency and stability, we can now address the convergence of the scheme. The main theorem is based on Lax, saying that for a linear numerical scheme, stability and conistency yield the convergence. In practice the theorem of Lax says that if we use a consistent scheme and we do not observe any non-physical oscillations, then the numerical solution is 'close' to the sought (unknown exact) solution.

Theorem 6.36 (Lax)**.** *Let $u(t,x)$ be a sufficiently smooth solution of the heat equation posed in Formulation 6.24. Let u_j^n the discrete solution obtained by a finite difference scheme using the initial condition $u_j^0 := u_0(x_j)$. We assume that the finite difference scheme is linear, a one-step scheme, consistent, and stable in the $\|\cdot\|$ norm. Then the scheme is convergent and it holds for all $T > 0$*

$$\lim_{h,k\to 0}\Big(\sup_{t_n\le T}\|e^n\|\Big) = 0$$

where $e_j^n := u_j^n - u(t_n, x_j)$ is the error vector. Moreover, the order of convergence is p in space and q in time. For all $T > 0$, there exists a constant $C_T > 0$ such that

$$\sum_{t_n\le T}\|e^n\| \le C_T\Big(h^p + k^q\Big).$$

Proof. We assume that we work with Dirichlet conditions. From before we know that

$$u^{n+1} = Au^n.$$

Let u be an exact solution of Formulation 6.24. We denote $v^n = (v_j^n)_{1\le j\le N}$ with $v_j^n = u(t_n, x_j)$. Since the scheme is consistent, there exist a vector ε^n such that

$$v^{n+1} = Av^n + k\varepsilon^n$$

with $\lim_{h,k\to 0}\|\varepsilon^n\| = 0$, where the convergence is uniformely for $0 \le t_n \le T$. Furthermore,

$$\|\varepsilon^n\| \le C(h^p + k^q).$$

We now use

$$e_j^n = u_j^n - u(t_n, x_j)$$

Subtraction of the perturbed iteration and exact iteration yields

$$e^{n+1} = Ae^n - k\varepsilon^n.$$

By recursion, we obtain

$$e^n = A^n e^0 - k\sum_{i=1}^{n} A^{n-i}\varepsilon^{i-1}.$$

We now argue with the stability

$$\|u^n\| = \|A^n u^0\| \le K\|u^0\|$$

i.e.,

$$\|A^n\| \le K$$

Since $e^0 = 0$, the recursion yields

$$\|e^n\| \le k\sum_{i=1}^{n}\|A^{n-i}\|\|\varepsilon^{i-1}\| \le knKC(h^p + k^q).$$

This was to show, where we recognize that $C_T := TKC$. □

6.9.6 Extended exercise

Let $T > 0$ and $\nu > 0$. We define the domains $\Omega := \mathbb{R}$ and $I := (0, T)$. Furthermore, let $u^0 \in C^0(\Omega)$ a 1-periodic function, i.e., $u^0(0,t) = u^0(1,t)$. Find $u(x,t) : \Omega \times I \to \mathbb{R}$ such that

$$\begin{aligned} \partial_t u(x,t) - \nu \partial_{xx} u(x,t) = 0 \quad &\text{in } \Omega \times I, \\ u(x,0) = u^0(x) \quad &\text{in } \Omega \times \{0\}. \end{aligned}$$

We assume that this probem is well-posed. In the following we are now interested in

1. Mass conservation: Show that for all $t \in I$, it holds

$$\int_0^1 u(x,t)\,dx = \int_0^1 u^0(x)\,dx.$$

2. Energy dissipation: Show that for all $t \in I$, it holds

$$\frac{1}{2}\int_0^1 |u(x,t)|^2\,dx \leq \frac{1}{2}\int_0^1 |u^0(x)|^2\,dx$$

3. Discretization with a θ-scheme in time (see also Section 12.3.2). Let $\theta \in [0,1]$. Let $N \in \mathbb{N}$ with $N \geq 1$. We define the time step size as $k = \frac{T}{N}$. For all $0 \leq n \leq N$, we introduce the discrete time points $t^n = nk$. For the spatial discretization, we work with $J \in \mathbb{N}$ with $J \geq 1$. The spatial mesh size is defined by $h = \frac{1}{J}$ (since 1-periodic, we only consider the interval $(0,1)$). We now introduce the discrete nodes $x_j = jh$ for $-1 \leq j \leq J+1$. The θ discretization of the heat equation yields:

$$\frac{U_j^{n+1} - U_j^n}{k} - \theta\nu\frac{U_{j+1}^{n+1} - 2U_j^{n+1} + U_{j-1}^{n+1}}{h^2} - (1-\theta)\nu\frac{U_{j+1}^n - 2U_j^n + U_{j-1}^n}{h^2} = 0$$

for all $0 \leq n \leq N-1$ and for all $1 \leq j \leq J$.

Task 1: Justify for yourself that the θ discretization makes sense and corresponds to the finite differences from classes introducing numerical methods

We now add the **boundary conditions**. Because of the 1-periodicity, we have:

$$\forall 0 \leq n \leq N-1, \quad U_0^n = U_J^n, \quad U_{J+1}^n = U_1^n.$$

It remains to pose the **initial conditions** at $n = 0$:

$$\forall 1 \leq j \leq J, \quad U_j^0 = u^0(x_j).$$

We introduce further

$$C_d = \nu\frac{k}{h^2}.$$

For all $0 \leq n \leq N$ we introduce the vector $U^n = (U_i^n)_{1 \leq i \leq J} \in \mathbb{R}^J$.

Task 2: Write the θ scheme in the following matrix form for $0 \leq n \leq N-1$:

$$(B_I^\theta)^{-1} U^{n+1} = B_E^\theta U^n$$

with

$$B_I^\theta = (I + \theta C_d B)^{-1}, \quad B_E^\theta = I - (1-\theta)C_d B$$

where $I \in \mathbb{R}^{J \times J}$ is the identity matrix and $B \in \mathbb{R}^{J \times J}$ a quadratic matrix (please explain its entries!).
Task 3: What happens to B_I^θ and B_E^θ when $\theta = 0$ or 1?
Task 4: Please justify furthermore the existence of B_I^θ.

4. Truncation error / convergence error. We introduce the vector $e^n \in \mathbb{R}^J$ with

$$e_j^n = U_j^n - V_j^n,$$

where the exact solution evaluated at the discrete time and spatial points is denoted by $V_j^n = (V_i^n)_{1\leq i\leq J} = (u(x_i, t^n))_{1\leq i\leq J} \in \mathbb{R}^J$. Because of the boundary conditions, we have

$$e_0^n = e_J^n \quad \text{and} \quad e_{J+1}^n = e_1^n$$

for $0 \leq n \leq N-1$ and because of $e_j^0 = 0$ for $1 \leq j \leq J$. Show that for $0 \leq n \leq N-1$, it holds

$$(B_I^\theta)^{-1} e^{n+1} = B_E^\theta e^n - k\eta^n.$$

Furthemore, please provide the details of the truncation error $\eta^n \in \mathbb{R}^J$.

5. Consistency. Show that the θ-schema is consistent with the order 1 in time and 2 in space. Please specify which order of the exact solution is necessary to obtain these (optimal) results. Finally, what happens for the choice $\theta = 0.5$?

6.10 Chapter summary and outlook

In this chapter, we considered finite differences for elliptic problems. Despite that finite elements or isogeometric analysis have become in general more important, finite differences still serve their purposes because they are simple to implement and do still exist in large industrial codes. FD are nevertheless further developed (mimetic finite differences, etc.), and are also often used in combination with other methods such as finite elements. For instance, a widely used discretization method for initial boundary value problems is to discretize in time with a finite difference scheme (e.g., One-Step-θ) and to discretize in space with a finite element method, which is the topic in Section 12. But let us first investigate tools from functional analysis, approximation theory, and FEM in the next two chapters.

7 Functional analysis and approximation theory

In this chapter, we gather all ingrediants from analysis, functinal analysis, and approximation theory in order to study PDEs from a theoretical point of view. Most results have immediate practical consequences on the development of good numerical schemes.

7.1 Analysis, linear algebra, functional analysis and Sobolev spaces

Functional analysis is the combination of classical analysis (i.e., calculus) and linear algebra. The latter one is introduced for finite dimensional spaces in first semester classes. In functional analysis, infinite dimensional vector spaces are considered, which require tools from analysis such as functions, operators, and convergence results. The extension to infinite dimensions leads to many nontrivial results. Classical books are [4, 28, 87, 134, 146].

We cannot provide a complete introduction, but only the most important concepts necessary for the further understanding of the present notes. At the beginning on the next two pages, we partially follow in the spirit of Tröltzsch [131][p. 17-19] since therein a nice, compact, introduction to the basic tools is provided. The analysis results (in particlar the Lebuesgue integral) can be studied in more detail with the help of Königsberger [86].

Before we begin, we refer the reader to Section 3.11 in which the definition of a normed space is recapitulated.

7.1.1 Convergence and complete spaces

Definition 7.1 (Convergence)**.** *A sequence $(u_n)_{n\in\mathbb{N}} \subset \{V, \|\cdot\|\}$ **converges**, if there exists a limit $u \in V$ such that*

$$\lim_{n\to\infty} \|u_n - u\| = 0$$

or in other notation

$$\lim_{n\to\infty} u_n = u.$$

Definition 7.2 (Cauchy sequence)**.** *A sequence is called a **Cauchy sequence** if for all $\varepsilon > 0$, there exists $N \in \mathbb{N}$ such that*

$$\|u_n - u_m\| \leq \varepsilon \quad \forall n > N, \quad \forall m > N$$

Any sequence that converges is a Cauchy sequence. The opposite is in general not true.

Example 7.3. *Let $\Omega = (0, 2)$. We consider the space $\{C(\Omega), \|\cdot\|_{L^2}\}$ with*

$$u_n(x) = \min\{1, x^n\}.$$

We see (check yourself)

$$\|u_n - u_m\|_{L^2}^2 = \int_0^2 (u_n - u_m)^2 \, dx = \ldots = \ldots \leq \frac{2}{2m+1}$$

for $m \leq n$. It is clear that this is a Cauchy sequence. However the limit u is not contained in the space $\{C(\Omega), \|\cdot\|_{L^2}\}$ because it is not a continuous function:

$$u(x) = \lim_{n\to\infty} u_n(x) = \begin{cases} 0, & 0 \leq x \leq 1 \\ 1, & 1 \leq x \leq 0 \end{cases}$$

A very important class of function spaces are those in which all Cauchy sequences do converge:

Definition 7.4 (Complete space, Banach space)**.** *A normed space $\{V, \|\cdot\|_V\}$ is a **complete space**, if all Cauchy sequences converge (i.e., the Cauchy sequence has a limit that belongs to V). A complete, normed space is a so-called **Banach space**.*

Example 7.5. *We continue our examples:*

1. *The set $\mathbb{Q}$ of rational numbers is* ***not*** *complete (e.g., [87]).*
2. *$\mathbb{R}^n$ with the euclidian norm $|x| = (\sum_{i=1}^n x_i^2)^{1/2}$ is a Banach space.*
3. *$C(\Omega)$ endowed with the maximum norm*

$$\|u\|_{C(\Omega)} = \max_{x\in\Omega} \|u(x)\|$$

is a Banach space.

4. *The space $\{C(\Omega), \|\cdot\|_{L^2}\}$ with*

$$\|u\|_{L^2} = \Big(\int_\Omega u(x)^2\, dx\Big)^{1/2}$$

is ***not*** *a Banach space; please see the previous example in which we have shown that the limit u does not belong to $C(\Omega)$ when we work with the L^2 norm.*

Remark 7.6. *The last example from before has already several consequences on our variational formulations. So far, we have mainly worked and proven our results using classical function spaces C, C^k, C^∞. On the other hand, we work with integrals, i.e, $(\nabla u, \nabla\phi) = \int_\Omega \nabla u \cdot \nabla\phi\, dx$. Observing that $\{C(\Omega), \|\cdot\|_{L^2}\}$ is not a Banach space and that the limit may be not contained in $C(\Omega)$ will influence wellposedness results and convergence results (recall that we try to proof later results similar as for finite differences).*

7.1.2 Scalar products, orthogonality, Hilbert spaces in 1D

In $\mathbb{R}^n$ a very important concept is associated to **orthogonality**, which does not come automatically with the Banach-space property. Those spaces in which a **scalar product** (inner product) can be defined, allow the definition of orthogonality. These spaces play a crucial role in PDEs and functional analysis.

Definition 7.7 (Pre-Hilbert space)**.** *Let V be a linear space (see Section 3.9). A mapping $(\cdot,\cdot) : V \to \mathbb{R}$ is called a* ***scalar product*** *if for all $u, v, w \in V$ and $\lambda \in R$ it holds*

$$\begin{aligned} (u,u) \geq 0 \quad \textit{and} \quad (u,u) = 0 \quad &\Leftrightarrow \quad u = 0 \\ (u,v) &= (v,u) \\ (u+v,w) &= (u,w) + (v,w) \\ (\lambda u, v) &= \lambda(u,v) \end{aligned}$$

If a scalar product is defined in V, this space is called a ***pre-Hilbert space****.*

Definition 7.8 (Hilbert space)**.** *A complete space endowed with an inner product is called a* ***Hilbert space****. The norm is defined by*

$$\|u\| := \sqrt{(u,u)}.$$

Example 7.9. *The space $\mathbb{R}^n$ from before has a scalar product and is complete, thus a Hilbert space. The space $\{C(\Omega), \|\cdot\|_{L^2}\}$ has a scalar product, but is not complete, and therefore not a Hilbert space. The space $\{C(\Omega), \|\cdot\|_{C(\Omega)}\}$ is complete, but the norm is not induced by a scalar product and is therefore not a Hilbert space, but only a Banach space.*

Specifically, for studying PDEs we provide the following further examples:

Definition 7.10 (The L^2 space in 1D)**.** *Let $\Omega = (a,b)$ be an interval (recall 1D Poisson). The space of square-integrable functions on Ω is defined by*

$$L^2(\Omega) = \{v : \int_\Omega v^2\, dx < \infty\}$$

The space L^2 is a Hilbert space equipped with the scalar product

$$(v,w) = \int_\Omega vw\, dx$$

and the induced norm

$$\|v\|_{L^2} := \sqrt{(v,v)}.$$

Using Cauchy's inequality

$$|(v,w)| \leq \|v\|_{L^2}\|w\|_{L^2},$$

we observe that the scalar product is well-defined when $v, w \in L^2$. A mathematically very correct definition must include in which sense (Riemann or Lebesgue) the integral exists. In general, all L spaces are defined in the sense of the Lebuesgue integral (see for instance [86][Chapter 7]).

Definition 7.11 (The H^1 space in 1D)**.** *We define the $H^1(\Omega)$ space with $\Omega = (a,b)$ as*

$$H^1(\Omega) = \{v : \ v \text{ and } v' \text{ belong to } L^2\}$$

This is space is equipped with the following scalar product:

$$(v,w)_{H^1} = \int_\Omega (vw + v'w')\,dx$$

and the norm

$$\|v\|_{H^1} := \sqrt{(v,v)_{H^1}}.$$

Definition 7.12 (The H_0^1 space in 1D)**.** *We define the $H_0^1(\Omega)$ space with $\Omega = (a,b)$ as*

$$H_0^1(\Omega) = \{v \in H^1(\Omega) : \ v(a) = v(b) = 0\}.$$

The scalar product is the same as for the H^1 space.

Remark 7.13. *With the help of the space H_0^1 we are now able to re-state the 1D Poisson problem. Rather than working with the space V (see (72)), we work with H_0^1. In fact this is the largest space that allows to define the 1D Poisson problem:*

$$\text{Find } u \in H_0^1(\Omega): \quad a(u,\phi) = l(\phi) \quad \forall \phi \in H_0^1(\Omega)$$

with

$$a(u,\phi) = (u',\phi'), \quad l(\phi) = (f,\phi).$$

We also recall Definition 3.23 of a dual space. For instance, the dual space of H^1 is the space H^{-1}.

Definition 7.14 (Notation)**.** *When there is no confusion with the domain, we often write*

$$L^2 := L^2(\Omega), \quad H^1 := H^1(\Omega)$$

and so forth.

7.1.3 Lebuesgue measures and integrals

To formulate variational formulations, we have worked with function spaces so far, mostly denoted by V, that contain integral information and classical continuous functions.

However, to enjoy the full beauty of variational formulations, the Riemann integral is too limited (see textbooks to analysis such as [86][Chapter 7]) and the Lebuesgue integral is usually used. The latter one does hold for a larger class of integrable functions in terms of **completeness**, which is the biggest advantage in comparison to the Riemann integral. In consequence, one also obtains deep results in exchanging limiting processes such as passing to the limit with sequences and integration (see Chapter 8 in [86]). For instance, when does

$$\int f\,dx = \lim_{k\to\infty} \int f_k\,dx \quad \text{for } f_k \to f$$

hold true?

Based on the Lebuesgue integral, we can then formulate function spaces with derivative information, so-called **Sobolev spaces**, which become **Hilbert spaces** when a scalar product can be defined, i.e., when the basic underlying space is the L_2 space. The key feature of the Lebesgue integral is that Lebuesgue-measurable functions are considered for integration [86][Chapter 7]. Here, the set of Lebuesgue measure zero plays an important role.

Definition 7.15 (Lebesgue measurable, [86], Chapter 7.5)**.** *A set $\Omega \subset \mathbb{R}^n$ is **Lebesgue measurable** (or in short **measurable**) when the characteristic function 1 can be integrated over Ω. The number*

$$V(\Omega) := V^{(n)}(\Omega) := \int_\Omega 1\, dx = \int_{\mathbb{R}^n} 1_\Omega\, dx$$

is the n-dimensional volume or Lebesgues measure of A. In the case $n = 1$, this is the length of the interval. For $n = 2$ it is the area. For $n = 3$ it is the classical three-dimensional volume. The empty set has Lebesgue measure zero.

Definition 7.16 (Zero set in the sense of Lebesgue, [86], Chapter 7.6)**.** *A set $N \subset \mathbb{R}^n$ is a Lebesgue-set of measure zero if N is measurable with measure zero: $V(N) = 0$.*

Example 7.17. *We give three examples:*

- *The set $\mathbb{Q}$ of rational numbers has Lebesgue measure zero in $\mathbb{R}$ (see for instance [86][Chapter 7.6]).*
- *In $\mathbb{R}^n$ all subsets $\mathbb{R}^s$ with dimension $s < n$ have Lebesgue measure zero. (see for instance [86][Chapter 7.6]).*

The second example has important consequences for PDEs: boundaries $\Gamma \subset \mathbb{R}^{n-1}$ of domains $\Omega \subset \mathbb{R}^n$ have Lebesgue measure zero. To illustrate this in $\mathbb{R}^1$, we have a third example

- *Consider $\mathbb{R}^1$ and $\Omega = [-1, 1]$ and define*

$$f(x) := \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \quad \text{and} \quad g(x) := \begin{cases} 1 & x > 0 \\ 0 & x \leq 0 \end{cases}$$

  The two functions f and g differ in only one point, which is a set of measure zero in the Lebesgue sense. They are viewed representing the same function.

Definition 7.18 (Almost everywhere, a.e.)**.** *Let $\Omega \subset \mathbb{R}^n$ be a Lebesgue-measurable open domain. Measurable functions are defined **almost everywhere**, in short **a.e.**, in Ω. When we change the function value on a subset of Lebuesgue measure zero, then we do not change the function value. In other words*

$$f(x) = g(x) \quad \text{almost everywhere in } \Omega$$

when there is $T \subset \Omega$ such that the Lebesgue measure of T is zero and

$$f(x) = g(x) \quad \text{for all } x \in \Omega \setminus T.$$

Short: Almost everywhere means except on a set of Lebesgue measure zero.

Definition 7.19 (L^2 space in $\mathbb{R}^n$)**.** *Let $\Omega \subset \mathbb{R}^n$ be a Lebesgue-measurable open domain. The space $L^2 = L^2(\Omega)$ contains all square-integrable functions in Ω:*

$$L^2 = \{v \text{ is Lebesgue measurable} \mid \int_\Omega v^2\, dx < \infty\}.$$

The space L^2 is complete (each Cauchy sequence converges in the norm defined below) and thus a Banach-space. For a proof see for instance [87][Section 2.2-7] or [134][Kapitel I]. Moreover, we can define a scalar product:

$$(u, v) = \int_\Omega vu\, dx$$

such that L^2 is even a Hilbert space with norm

$$\|u\|_{L^2} = \sqrt{(u, u)}.$$

In the same way, one obtains all L^p spaces:

Definition 7.20 (L^p spaces)**.** *Let $1 \leq p < \infty$. Then,*

$$L^p = \{v \text{ is Lebesgue measurable} \mid \int_\Omega v^p \, dx < \infty\}$$

equipped with the norm:

$$\|u\|_{L^p} := \Big(\int_\Omega |v|^p \, dx\Big)^{\frac{1}{p}}.$$

For $p = \infty$, we define the norm via:

$$\|u\|_{L^\infty} := ess\sup |u|.$$

which are the essentially bounded functions.

Proposition 7.21. *Let Ω be bounded. Then:*

$$L^\infty \subset \ldots \subset L^p \subset L^{p-1} \subset \ldots \subset L^2 \subset L^1.$$

We recall several important results from analysis that have an important impact on variational formulations of PDEs and numerics. The first result is:

Definition 7.22 (Density, e.g., [87])**.** *A metric space U is **dense** in V when*

$$\bar{U} = V.$$

Here $\bar{U}$ means the closure of U. Every point of V is either in U or arbitrarly close to a member of U.

Example 7.23. *Two examples:*

- *The set $\mathbb{Q}$ of rational numbers is dense in $\mathbb{R}$ (e.g., [87])*
- *The space of polynomial functions P_k is dense in $C[a,b]$ (see lectures on Analysis 1, e.g. [85]. In other words: to each $f \in C[a,b]$, there exists a sequence $(P_k)_{k\in\mathbb{N}}$ of polynomials, which converges uniformely to f. This is well known as the Weierstrass approximation theorem*

Theorem 7.24 (Density result for L^2)**.** *The space $C_c^\infty(\Omega)$ is **dense** in $L^2(\Omega)$. In other words: for all functions $f \in L^2$, there exists a sequence $(f_n)_{n\in\mathbb{N}} \subset C_c^\infty(\Omega)$ such that*

$$\lim_{n\to\infty} \|f - f_n\| = 0.$$

Remark 7.25. *The previous result means that we can approximate any L^2 function by a sequence of 'very nice', smooth, functions defined in the usual classical sense.*

Remark 7.26 (Density results)**.** *In analysis and functional analysis, density arguments are often adopted since it allows to work with simple functions in order to approximate 'abstract' and complicated functions.*

As second result, we generalize the fundamental lemma of calculus of variations:

Theorem 7.27 (Fundamental lemma of calculus of variations)**.** *Let $f \in L^2(\Omega)$. If*

$$\int_\Omega f\phi \, dx = 0 \quad \forall \phi \in C_c^\infty(\Omega),$$

then $f(x) = 0$ almost everywhere in Ω.

Proof. For $f \in L^2$ according to the density result, there exists a sequence $(f_n)_{n\in\mathbb{N}} \subset C_c^\infty(\Omega)$ with

$$\|f_n - f\| \to 0 \quad (n \to \infty).$$

This yields

$$\|f\|_{L^2}^2 = \int_\Omega f^2 \, dx = (f,f) = (f,f) - \underbrace{(f,f_n)}_{=0 \text{ assumpt.!}} = (f, f - f_n) \leq \|f\|_{L^2}\|f - f_n\|_{L^2} \to 0 \quad \text{for } n \to \infty.$$

Therefore, $\|f\|_{L^2}^2 = 0$ if and only if $f = 0$ a.e..

□

Remark 7.28 (Historical note)**.** *The first version of the fundamental lemma of calculus of variations appeared around 1755/1756 when Lagrange found a solution to $\int_x^z F(x,y,y')\,dx = \min$ and Euler requested a more precise proof from him. Euler than gave in 1756 the name 'Variational calculus' in honor to Lagrange.*

7.1.4 Weak derivatives

The definition of the space L^2 now allows us to define a more general derivative expression, the so-called **weak derivative**. In fact, this is the form of derivatives that we have implicitly used so far in our variational formulations.

Definition 7.29 (Weak derivative)**.** *Let $\Omega \subset \mathbb{R}^n$ be an open, measurable, domain. Let $u \in L^2$. We say that u is **weakly differentiable** in L^2, when functions $w_i \in L^2, i = 1, \dots, n$ exist such that for all functions $\phi \in C_c^\infty(\Omega)$, we have*

$$\int_\Omega u \partial_{x_i} \phi \, dx = -\int_\Omega w_i(x)\phi(x) \, dx.$$

In our usual compact form:

$$(u, \partial_{x_i}\phi) = -(w_i, \phi).$$

Each w_i is the ith partial derivative of u. Thus

$$\partial_{x_i} u = w_i \quad \text{in the weak sense.}$$

We notice that there are no boundary terms since φ has compact support in Ω and therefore vanishes near the boundary $\partial\Omega$. We also recognize that the weak derivative is nothing else than again integration by parts

Remark 7.30 (Example)**.** *As the terminology indicates, the weak derivative is weaker than the usual (strong) derivative expression. In classical function spaces both coincide.*

Example 7.31 ([131])**.** *Consider $u(x) = |x|$ in $\Omega = (-1, 1)$. Then, the weak derivative is given by:*

$$u'(x) := w(x) = \begin{cases} -1 & x \in (-1, 0) \\ +1 & x \in [0, 1) \end{cases}.$$

Indeed it holds that for all $v \in C_c^\infty(\Omega)$ we have

$$\int_{-1}^{1} |x| v'(x) \, dx = \int_{-1}^{0} (-x) v'(x) + \int_0^1 x v'(x) \, dx = \ldots = -\int_{-1}^{1} w(x) v(x) \, dx.$$

We notice that it is not important which value u' has in zero since it is a point of measure zero; see Example 7.17.

Example 7.32 ([51])**.** *Let $n = 1$ and $\Omega = (0, 2)$ and*

$$u(x) = \begin{cases} x & \text{if } \ 0 < x \leq 1 \\ 2 & \text{if } \ 1 < x < 2. \end{cases}$$

The function u' has no weak derivative! There does not exist any function $v \in L^1_{loc}$ that satisfies the relation

$$\int_0^2 u\varphi' \, dx = \int_0^2 v\varphi \, dx \quad \forall \varphi \in C_c^\infty(\Omega).$$

We proof by a counter example. Suppose the weak derivative exists for v and all φ. Then:

$$-\int_0^2 v\varphi \, dx = \int_0^2 u\varphi' \, dx = \int_0^1 x\varphi' \, dx + 2\int_1^2 \varphi' \, dx = -\int_0^1 \varphi \, dx - \varphi(1)$$

We choose a sequence of test functions $(\varphi_m)_{m\in\mathbb{N}}$ with

$$0 \leq \varphi_m \leq 1, \quad \varphi_m(1) = 1, \quad \varphi_m(x) \to 0$$

for all $x \neq 1$. We now take $\varphi := \varphi_m, m \to \infty$, we obtain

$$1 = \lim_{m\to\infty} \varphi_m(1) = \lim_{m\to\infty} \left[\int_0^2 v\varphi_m \, dx - \int_0^1 \varphi_m \, dx \right] = 0.$$

This is a contradiction.

We can simply extend the weak derivative to higher-order derivatives. Here we remind the reader to the multiindex notation presented in Section 3.6. It holds:

Definition 7.33. *Let $u \in C^k(\bar{\Omega})$ and $v \in C_c^k(\Omega)$ and α a multiindex with $|\alpha| \leq k$. Then employing multiple times partial integration, we obtain*

$$\int_\Omega u(x) D^\alpha v(x)\, dx = (-1)^{|\alpha|} \int_\Omega v(x) D^\alpha u(x)\, dx.$$

Definition 7.34 (Higher-order weak derivative)**.** *Let $\Omega \subset \mathbb{R}^n$ be an open, measurable, domain. Let u be locally integrable in the sense of Lebesgue and α be a multiindex. If there exists a locally integrable function w with*

$$\int_\Omega u(x) D^\alpha v(x)\, dx = (-1)^{|\alpha|} \int_\Omega v(x) w(x)\, dx,$$

then $w = D^\alpha u(x)$ and the so-called weak derivative (of higher-order).

7.1.5 The Hilbert spaces H^1 and H_0^1

With the help of weak derivatives, we now define Lebuesgue spaces with (weak) derivatives: **Sobolev spaces.** When their norms are induced by a scalar product, namely L_2-based spaces, a Sobolev space is a **Hilbert space**.

Definition 7.35 (The space H^1)**.** *Let $\Omega \subset \mathbb{R}^n$ be an open, measurable, domain. The space $H^1 := H^1(\Omega)$ is defined by*

$$H^1 := \{v \in L^2(\Omega) |\ \partial_{x_i} v \in L^2, \quad i = 1, \ldots, n\}$$

In compact form:

$$H^1 := \{v \in L^2(\Omega) |\ \nabla v \in L^2\}.$$

The nabla-operator has been defined in Section 3.7.

Remark 7.36. *In physics and mechanics, the space H^1 is also called the **energy space** . The associated norm is called **energy norm**.*

Proposition 7.37 (H^1 is a Hilbert space)**.** *We define the scalar product*

$$(u, v)_{H^1} := \int_\Omega (uv + \nabla u \cdot \nabla v)\, dx$$

which induces the norm:

$$\|u\|_{H^1} = \sqrt{(u, u)_{H^1}}.$$

The space H^1 equipped with the norm $\|u\|_{H^1}$ is a Hilbert space.

Proof. It is tivial to see that $(u, v)_{H^1}$ defines a scalar product. It remains to show that H^1 is complete. Let $(u_n)_{n\in N}$ be a Cauchy sequence in H^1. We need to show that this Cauchy sequence converges in H^1. Specifically $(u_n)_{n\in N}$ and $(\partial_{x_i} u_n)_{n\in N}$ are Cauchy sequences in L^2. Since L^2 is complete, see Definition 7.19, there exist two limits u and w_i with

$$\begin{aligned} u_n &\to u \quad \text{in } L^2 \quad \text{for } n \to \infty \\ \partial_{x_i} u_n &\to w_i \quad \text{in } L^2 \quad \text{for } n \to \infty. \end{aligned}$$

We employ now the weak derivative:

$$\int_\Omega u_n(x) \partial_{x_i} \phi(x)\, dx = -\int_\Omega \partial_{x_i} u_n(x) \phi(x)\, dx \quad \phi \in C_c^\infty$$

Passing to the limit $n \to \infty$ yields

$$\int_\Omega u(x) \partial_{x_i} \phi(x)\, dx = -\int_\Omega w_i(x) \phi(x)\, dx.$$

This shows that u has a weak derivative w. Therefore, the element w is contained in H^1 and $(u_n)_{n\in N} \subset H^1$, which shows the assertion. □

Remark 7.38. *The space H^1 is the closure of $C^1(\bar{\Omega})$ with respect to the H^1 norm. It also holds for Ω that is regular, open, bounded that $C_c^\infty(\bar{\Omega})$ is dense in $H^1(\Omega)$.*

7.1.6 Hilbert spaces versus classical function spaces

Even so we can approximate Lebuesgue-measurable functions in L^2 or H^1 with functions from $C_c^\infty(\Omega)$, the limit function may not have nice properties. The most well-known example is the following:

Proposition 7.39 (Singularities of H^1 functions)**.** *Let $\Omega \subset \mathbb{R}^n$ be open and measurable. It holds:*

- *For $n = 1$, $H^1(\Omega) \subset C(\Omega)$.*
- *For $n \geq 2$, functions in $H^1(\Omega)$ are in general neither continuous nor bounded.*

Proof. Let $n = 1$. Let $\Omega = [a, b]$ and $C^\infty(\Omega)$. Then, we have for $|x - y| < \delta$ and using Cauchy-Schwarz

$$|u(x) - u(y)| = \int_x^y v'(t)\,dt \leq \Big(\int_x^y 1^2\,dt\Big)^{1/2}\Big(\int_x^y |v'(t)|^2\,dt\Big)^{1/2} \leq \sqrt{\delta}\|v\|_{H^1}.$$

Consequently, each Cauchy sequence in $H^1 \cap C^\infty$ is equicontinuous and bounded. Using Arcelà-Ascoli (for a general version we refer to [134]), the limit function is continuous.

Now let $n \geq 2$. Let the unit circle be defined as

$$D := \{(x, y) \in \mathbb{R}^2 |\, x^2 + y^2 < 1\}.$$

We define the function

$$u(x, y) = \log\log\frac{2}{r}$$

where r is the radius with $r^2 = x^2 + y^2$. It is obvious that u is unbounded. On the other hand, $u \in H^1$ because

$$\int_0^{1/2} \frac{1}{r\log^2 r}\,dr < \infty.$$

For $n \geq 3$, we have

$$u(x) = r^{-\alpha}, \quad \alpha < (n - 2)/2.$$

It holds $u \in H^1$ with a singularity in the point of origin. It seems that the higher n is, the more significant the singularities are. □

Definition 7.40 (H_0^1)**.** *The space $H_0^1(\Omega)$ contains vanishing function values on the boundary $\partial\Omega$. The space $H_0^1(\Omega)$ is the closure of $C_c^\infty(\Omega)$ in H^1.*

One often writes:

Definition 7.41 (H_0^1)**.** *The space H_0^1 (spoken: 'H,1,0' and **not** 'H,0,1') is defined by:*

$$H_0^1(\Omega) = \{v \in H^1 |\, v = 0 \text{ on } \partial\Omega\}.$$

7.1.7 Sobolev spaces

Lebuesgue spaces L^p spaces with $p \neq 2$ do not have scalar products, but are of course still complete. Introducing weak derivatives yield the so-called **Sobolev spaces**. For $p = 2$ a Sobolev space is a Hilbert space.

Definition 7.42 (Higher-order Sobolev spaces)**.** *Let $1 \leq p < \infty$ and $k \in \mathbb{N}$. The normed space $W^{k,p}(\Omega)$ contains all functions $v \in L^p$, which derivatives $D^\alpha v$ with $|\alpha| \leq k$ also belong to L^p. The norm is defined by*

$$\|v\|_{W^{k,p}} = \Big(\sum_{|\alpha|\leq k}\int_\Omega |D^\alpha v(x)|^p\Big)^{1/p}.$$

For $p = \infty$ we define

$$\|v\|_{W^{k,\infty}} = \max_{|\alpha|\leq k}\|D^\alpha v\|_{L^\infty}.$$

The spaces $W^{k,p}$ are Banach spaces. For $p = 2$, we obtain Hilbert spaces. For instance choosing $k = 1$ and $p = 2$ yields

$$H^1 := W^{1,2}.$$

7.1.8 Brief summary of L^2, H^1 and H^2

Let $\Omega \subset \mathbb{R}^n$ be a Lebesgue measurable domain.

- $L^2(\Omega)$ space:
$$L^2 = \{v \text{ is Lebesgue measurable } | \int_\Omega v^2 \, dx < \infty\}.$$
- L^2 scalar product:
$$(u, v)_{L^2} := \int_\Omega u \cdot v \, dx.$$
- L^2 norm:
$$\|u\|_{L^2} := \sqrt{(u, u)_{L^2}} := \int_\Omega |u|^2 \, dx.$$
- $H^1(\Omega)$ space:
$$H^1 = \{v \in L^2(\Omega) | \ \partial_{x_i} v \in L^2(\Omega), \ i = 1, \ldots, n\}.$$
- H^1 scalar product:
$$(u, v)_{H^1} := \int_\Omega [u \cdot v + \nabla u \cdot \nabla v] \, dx.$$
- H^1 norm:
$$\|u\|_{H^1} := \sqrt{(u, u)_{H^1}} := \int_\Omega [|u|^2 + |\nabla u|^2] \, dx.$$
- H^1 seminorm (not a norm, because in general we do not have $|u|_{H^1} \Rightarrow u = 0$):
$$|u|_{H^1} := \sqrt{(\nabla u, \nabla u)_{L^2}} := \int_\Omega |\nabla u|^2 \, dx.$$
- $H^2(\Omega)$ space:
$$H^2 = \{v \in L^2(\Omega) | \ \partial_{x_i} v \in L^2(\Omega), \ \partial_{x_i x_j} v \in L^2(\Omega), \ i, j = 1, \ldots, n\}.$$
- H^2 scalar product:
$$(u, v)_{H^2} := \int_\Omega [u \cdot v + \nabla u \cdot \nabla v + \nabla^2 u : \nabla^2 v] \, dx.$$
- H^2 norm:
$$\|u\|_{H^2} := \sqrt{(u, u)_{H^2}} := \int_\Omega [|u|^2 + |\nabla u|^2 + |\nabla^2 u|^2] \, dx.$$
- H^2 seminorm (not a norm, because in general we do not have $|u|_{H^2} \Rightarrow u = 0$):
$$|u|_{H^2} := \sqrt{(\nabla^2 u, \nabla^2 u)_{L^2}} := \int_\Omega |\nabla^2 u|^2 \, dx.$$

7.2 Useful Inequalities

The Hölder inequality reads:

Proposition 7.43 (Hölder's inequality)**.** *Let $1 \leq p \leq \infty$ and $\frac{1}{p} + \frac{1}{q} = 1$ (with the convention $\frac{1}{\infty} = 0$). Let $f \in L^p$ and $g \in L^q$. Then,*

$$fg \in L^1,$$

and

$$\|fg\|_{L^1} \leq \|f\|_{L^p} \|g\|_{L^q}.$$

Proof. See Werner [134]. □

Corollary 7.44 (Cauchy-Schwarz inequality)**.** *Set $p = q = 2$. Then*

$$\|fg\|_{L^1} \leq \|f\|_{L^2} \|g\|_{L^2}.$$

Proposition 7.45 (Minkowski's inequality)**.** *Let $1 \leq p \leq \infty$ and let $f \in L^p$ and $g \in L^p$. Then,*

$$\|f + g\|_{L^p} \leq \|f\|_{L^p} + \|g\|_{L^p}.$$

Proposition 7.46 (Cauchy's inequality with ε)**.**

$$ab \leq \varepsilon a^2 + \frac{b^2}{4\varepsilon}$$

for $a, b > 0$ and $\varepsilon > 0$. For $\epsilon = \frac{1}{2}$, the original Cauchy inequality is obtained.

Proposition 7.47 (Young's inequality)**.** *Let $1 < p, q < \infty$ and $\frac{1}{p} + \frac{1}{q} = 1$. Then,*

$$ab \leq \frac{a^p}{p} + \frac{b^q}{q}, \quad a, b > 0.$$

7.2.1 Some compactness results

A fundamental result in prooving well-posedness of PDEs in infinite dimensional spaces (recall that boundedness and closetness are not equivalent to compactness for infinite-dimensional spaces) is the following result:

Theorem 7.48 (Rellich)**.** *Let Ω be an open, bounded domain of class C^1. The embedding of H^1 into L^2 is compact. In other words: from each bounded sequence $(u_n) \subset H^1$, we can extract a subsequence $(u_{n_l}) \subset L^2$ with a limit in L^2.*

Proof. See for instance Wloka [145]. □

We have seen that H^1 in dimension $n \geq 2$ are not embedded into classical function spaces. However, higher-order Sobolev spaces have continuous representations. These are the so-called Sobolev embedding theorems. One result is

Proposition 7.49. *Let $\|u\|_\infty$ be the supremum norm. Let $\Omega \subset \mathbb{R}^2$ be convex polygonal or with Lipschitz boundary. Then:*

$$H^2(\Omega) \subset\subset C(\bar{\Omega})$$

and with $c > 0$, it holds

$$\|u\|_\infty \leq c\|u\|_{H^2} \quad \text{for } u \in H^2(\Omega)$$

Moreover, for each open and connected domain $\Omega \subset \mathbb{R}^2$, it holds:

$$H^2(\Omega) \subset\subset C(\Omega).$$

7.3 Approximation theory emphasizing on the projection theorem

The best approximation is based on findings from **approximation theory**, which we briefly recapitulate in the following.

We recall the main ingredients of orthogonal projections as they yield the best approximation properties.

7.3.0.1 Best approximation

Definition 7.50. *Let $U \subset X$ be a subset of a normed space X and $w \in X$. An element $v \in U$ is called best approximation to w in U if*

$$\|w - v\| = \inf_{u \in U} \|w - u\| \ ,$$

i.e., $v \in U$ has the smallest distance to w.

Proposition 7.51. *Let U be a finite-dimensional subspace of a normed space X. Then, there exists for each element in X a best approximation in U.*

Proof. Let $w \in X$. We choose a minimizing sequence (u_n) from U to approximate w. This fulfills:

$$\|w - u_n\| \to d \quad \text{with } n \to \infty$$

where $d := \inf_{u \in U} \|w - u\|$. Since

$$\begin{aligned} \|u_n\| &= \|u_n\| - \|w\| + \|w\| \\ &\leq | \ \|u_n\| - \|w\| \ | + \|w\| \\ &= | \ \|w\| - \|u_n\| \ | + \|w\| \\ &\leq \|w - u_n\| + \|w\| \end{aligned}$$

the sequence (u_n) is bounded. Because U is a finite-dimensional normed space, there exists a converging subsequence $(u_{n(l)})$ with

$$\lim_{l \to \infty} u_{n(l)} \to v \in U.$$

It follows that

$$\|w - v\| = \lim_{l \to \infty} \|w - u_{n(l)}\| = d.$$

□

Definition 7.52. *A pre-Hilbert space is a linear space with a scalar product.*

Proposition 7.53. *Let U be a linear subspace of a pre-Hilbert space X. An element $v \in U$ is a best approximation to $w \in X$ if and only if*

$$(w - v, u) = 0, \quad \text{for all } u \in U,$$

i.e., $w - v \perp U$. For each $w \in X$, we have at most one best approximation w.r.t. U.

Proof. " $\Leftarrow$ ":

Let $\|a\| := \sqrt{(a, a)}$, which holds for all $v, u \in U$:

$$\begin{aligned} \|w - u\|^2 &= \|w - v + v - u\|^2 \\ &= ((w - v) + (v - u), (w - v) + (v - u)) \\ &= (w - v, w - v) + (w - v, v - u) + (v - u, w - v) + (v - u, v - u) \\ &= \|w - v\|^2 + (w - v, v - u) + \overline{(w - v, v - u)} + \|v - u\|^2 \\ &= \|w - v\|^2 + 2 \cdot Re((w - v, v - u)) + \|v - u\|^2. \end{aligned}$$

Since U is a linear space, we have $v - u \in U$ and we require $(w - v, u) = 0 \forall u \in U$, it holds:

$$\begin{aligned} & \|w - u\|^2 = \|w - v\|^2 + \|v - u\|^2 \\ \Rightarrow \quad & \|w - v\|^2 = \|w - u\|^2 - \|v - u\|^2 \ , \quad \forall u \in U \\ \Rightarrow \quad & \|w - v\| < \|w - u\| \ , \quad \forall u \in U, u \neq 0 \end{aligned}$$

Therefore,

$$\|w-v\| = \inf_{u\in U} \|w-u\|$$

and thus v is a best approximation to w in U.

" $\Rightarrow$ ":

Let v be a best approximation to w in U. We assume that $(w-v,u_0) \neq 0$ for a $u_0 \in U$. Let us assume further that $(w-v,u_0) \in \mathbb{R}$ because U is a linear subspace of X. We choose $u = v + \frac{(w-v,u_0)}{\|u_0\|^2}u_0$ and conclude

$$\begin{aligned}
\|w-u\|^2 &= \|w-v\|^2 + 2Re\left(w-v, v-\left(v+\frac{(w-v,u_0)}{\|u_0\|^2}u_0\right)\right) \\
&\quad + \left\|v-\left(v+\frac{(w-v,u_0)}{\|u_0\|^2}u_0\right)\right\|^2 \\
&= \|w-v\|^2 + 2Re\left(w-v, -\frac{(w-v,u_0)}{\|u_0\|^2}u_0\right) + \left\|-\frac{(w-v,u_0)}{\|u_0\|^2}u_0\right\|^2 \\
&= \|w-v\|^2 + 2\left(-\frac{(w-v,u_0)}{\|u_0\|^2}Re(w-v,u_0)\right) + \frac{(w-v,u_0)^2}{\|u_0\|^4}\|u_0\|^2 \\
&= \|w-v\|^2 - 2\frac{(w-v,u_0)^2}{\|u_0\|^2} + \frac{(w-v,u_0)^2}{\|u_0\|^2} \\
&= \|w-v\|^2 - \frac{(w-v,u_0)^2}{\|u_0\|^2} \\
&< \|w-v\|^2.
\end{aligned}$$

This is a contradiction that v is a best approximation to w in U.

Uniqueness

Assume that v_1, v_2 are best approximations. Then,

$$(w-v_1, v_1-v_2) = 0 = (w-v_2, v_1-v_2), \quad \text{da } v_1-v_2 \in U.$$

and

$$\begin{aligned}
&\Rightarrow \quad (w,v_1-v_2) - (v_1,v_1-v_2) = (w,v_1-v_2) - (v_2,v_1-v_2) \\
&\Rightarrow \quad (v_1,v_1-v_2) = (v_2,v_1,-v_2) \\
&\Rightarrow \quad (v_1,v_1-v_2) - (v_2,v_1-v_2) = 0 \\
&\Rightarrow \quad (v_1-v_2,v_1-v_2) = 0 \\
&\Rightarrow \quad v_1-v_2 = 0 \\
&\Rightarrow \quad v_1 = v_2.
\end{aligned}$$

Thus, the best approximation is unique. □

Proposition 7.54 (Orthogonal projection)**.** *Let U be a complete linear subspace of a pre-Hilbert space X. Then, there exists for each $w \in X$ a unique best approximation in U. The operator $P : X \to U$, which maps each $w \in X$ on its best approximation, is a bounded linear operator. The properties of P are:*

$$P^2 = P \quad \textit{and} \quad \|P\| = 1$$

This is called an ***orthogonal projection*** *of X on U.*

Proof. We choose a sequence (u_n) with

$$\|w-u_n\|^2 \leq d^2 + \frac{1}{n}, \quad n \in \mathbb{N}$$

with $d := \inf_{u\in U} \|w-u\|$. Then:

$$\begin{aligned}
&\|(w-u_n)+(w-u_m)\|^2 + \|u_n-u_m\|^2 \\
&= ((w-u_n)+(w-u_m),(w-u_n)+(w-u_m)) \\
&\quad + ((u_n-w)+(w-u_m),(u_n-w)+(w-u_m)) \\
&= (w-u_n,w-u_n)+(w-u_n,w-u_m)+(w-u_m,w-u_n)+(w-u_m,w-u_m) \\
&\quad + (u_n-w,u_n-w)+(u_n-w,w-u_m)+(w-u_m,u_n-w)+(w-u_m,w-u_m) \\
&= 2\|w-u_n\|^2 + 2\|w-u_m\|^2 \\
&\leq 4d^2 + \frac{2}{n} + \frac{2}{m} \quad \text{for all } n,m\in\mathbb{N}.
\end{aligned}$$

Since $\frac{1}{2}(u_n+u_m)\in U$, it holds

$$\|w-\frac{1}{2}(u_n+u_m)\|^2 \geq d^2$$

and we can estimate $\|u_n-u_m\|^2$ via

$$\begin{aligned}
\|u_n-u_m\|^2 &\leq 4d^2 + \frac{2}{n} + \frac{2}{m} - \|(w-u_n)+(w-u_m)\|^2 \\
&= 4d^2 + \frac{2}{n} + \frac{2}{m} - \|2(w-\frac{1}{2}(u_n+u_m))\|^2 \\
&= 4d^2 + \frac{2}{n} + \frac{2}{m} - 4\|w-\frac{1}{2}(u_n+u_m)\|^2 \\
&\leq \frac{2}{n} + \frac{2}{m}.
\end{aligned}$$

Therefore, (u_n) is bounded and therefore a Cauchy sequence.

Since U is complete, there exists a $v\in U$ with $u_n \to v,\ n\to\infty$. From

$$\|w-u_n\|^2 \leq d^2 + \frac{1}{n}$$

it follows that $\lim_{n\to\infty} u_n = v$ is a best apprioximation to w in U. The uniqueness follows from Proposition (7.53).

Next, we show $P^2 = P$. Always, it holds $P(u) = u$. It follows

$$P^2(w) = P(P(w)) = P(u) = u = P(w)$$

The linearity of P is shown as follows: let $\alpha \neq 0$, then

$$\begin{aligned}
& P(\alpha x) = v \\
\Leftrightarrow\quad & (\alpha x - v, u) = 0 \\
\Leftrightarrow\quad & \alpha(x-\frac{v}{\alpha},u) = 0 \\
\Leftrightarrow\quad & P(x) = \frac{v}{\alpha} \\
\Leftrightarrow\quad & \alpha P(x) = v.
\end{aligned}$$

Consequently, $P(\alpha x) = \alpha P(x)$.

Let v_1, v_2 be best approximations of x, y, i.e., it holds $P(x) = v_1, P(y) = v_2$. Then:

$$\begin{aligned}
& P(x)+P(y) = v_1+v_2 \\
\Leftrightarrow\quad & (x-v_1,u) = 0 \quad\wedge\quad (y-v_2,u) = 0 \\
\Leftrightarrow\quad & (x-v_1,u)+(y-v_2,u) = 0 \\
\Leftrightarrow\quad & (x-v_1+y-v_2,u) = 0 \\
\Leftrightarrow\quad & ((x+y)-(v_1+v_2),u) = 0.
\end{aligned}$$

Thus, $P(x+y) = v_1 + v_2 = P(x) + P(y)$. It remains to show that $\|P\| = 1$. It holds:

$$\begin{aligned}\|w\|^2 &= (P(w) + w - P(w), P(w) + w - P(w)) \\ &= (P(w), P(w)) + (w - P(w), w - P(w)) \\ &\quad + (P(w), w - P(w)) + (w - P(w), P(w)) \\ &= \|P(w)\|^2 + \|w - P(w)\|^2 \\ &\geq \|P(w)\|^2 \quad \forall w \in X\end{aligned}$$

Thus, P is bounded with $\|P\| \leq 1$. Because $P^2 = P$ and since for two lineaer operators A and B, it holds $\|AB\| \leq \|A\|\|B\|$, it follows finally

$$\begin{aligned}& \|P\| = \|P^2\| = \|PP\| \leq \|P\|\|P\| \\ \Rightarrow \quad & \|P\| \geq 1 \\ \Rightarrow \quad & \|P\| = 1.\end{aligned}$$

□

7.3.0.2 The Fréchet-Riesz representation theorem An important principle in functional analysis is to gain information of normed spaces with the help of functionals.

7.3.0.2.1 Direct sum

Definition 7.55 (Direct sum)**.** *A vector space X can be written as direct sum of two subspaces Y and Z:*

$$X = Y \oplus Z,$$

if each element $x \in X$ has a unique representation such as

$$x = y + z, \quad y \in Y,\, z \in Z$$

Remark 7.56. *The space Z is the **algebraic complement** of Y in X.*

In Hilbert space theory we are specifically interested in such representations. A general Hilbert space H can be written as direct sum of a closed subspace Y and its orthogonal complement $Y^\perp$.

Proposition 7.57 (Projection theorem, direct sum)**.** *Let Y be a closed subspace of a Hilbert space H. Then:*

$$H = Y \oplus Z, \quad Z = Y^\perp$$

7.3.0.2.2 The Fréchet-Riesz representation theorem

Definition 7.58. *The space $\mathcal{L} = L(X, \mathbb{K})$ of the linear, bounded functionals on a normed space is called the dual space of X. We often find the notations $X' = \mathcal{L} = L(X, \mathbb{K})$ or if X is a Hilbert space, we write $X^* = \mathcal{L} = L(X, \mathbb{K})$.*

One of the key theorems of Hilbert space theory is:

Proposition 7.59. *Let X be a Hilbert space. For each linear, bounded functional $F \in \mathcal{L}$, there exists a unique element $f \in X$, so that*

$$F(u) = (u, f) \qquad \forall u \in X.$$

We have constructed a bijective, isogeometric conjugate linear mapping with

$$\|f\| = \|F\|.$$

Proof. We divide the proof into several parts:

- **Uniqueness**
  When f maps on $F = 0$, then
  $$F(x) = (x, f) = 0 \qquad \forall x \in X$$
  and also
  $$(f, f) = 0$$
  from which follows
  $$f = 0.$$
  Thus, the mapping $f \to F$ is unique because $f = 0$ is the only element, which induces the zero function $F = 0$.

- **Equality of norms**
  Cauchy-Schwarz yields
  $$\begin{aligned} & ||F|| = |F| = |(x, f)| \leq ||x|| \cdot ||f|| \\ \Rightarrow \qquad & \frac{|F|}{||x||} \leq ||f|| \\ \Leftrightarrow \qquad & \sup_{||x|| \leq 1} \frac{|F(x)|}{||x||} \leq ||f|| \\ \Rightarrow \qquad & ||F|| \leq ||f|| \end{aligned}$$
  with
  $$||F|| = \sup_{||x|| \leq 1} \frac{|F(x)|}{||x||}$$
  We use specifically $x = f$ in $F(x) = (x, f)$ and obtain:
  $$|(f, f)| = ||f||^2 = |F(f)| \leq ||f|| \cdot ||F||.$$
  Therefore:
  $$||f|| \leq ||F||$$
  and therefore, we have on the one hand $||f|| \geq ||F||$. On the other hand, we have $||f|| \leq ||F||$. From this it follows:
  $$||f|| = ||F||$$

- **Construction**
  For $F \in X^* = \mathcal{L}$ we need to construct $f \in X$:
  The kernel
  $$N(F) = \{u \in X : F(u) = 0\}$$
  is a closed, linear subspace of the Hilbert space X. If $N = X$, then $f = 0$ yields the desired result. If $N \neq X$, then it holds
  $$X = N \oplus N^{\perp}$$
  We choose now a $w \in X$ with $F(w) \neq 0$. From the projection theorem it follows that if v is best approximation of w onto the subspace N, then
  $$w - v \perp N(F)$$
  It holds with $g := w - v$,
  $$F(g)u - F(u)g \quad \in N(F) \qquad \forall u \in X$$
  because
  $$F(F(g)u - F(u)g) \; = \; F(g)F(u) - F(u)F(g) \; = \; 0.$$

Then:

$$\begin{aligned} & (F(g)u - F(u)g, g) &=& \ 0 \qquad \forall u \in X \\ \Rightarrow \quad & (F(g)u, g) - (F(u)g, g) &=& \ 0 \\ \Rightarrow \quad & (F(u)g, g) &=& \ (F(g)u, g) \\ \Rightarrow \quad & F(u)(g, g) &=& \ (u, \overline{F(g)}g) \\ \Rightarrow \quad & F(u) &=& \ \left(\frac{u, \overline{F(g)}g}{||g||^2}\right) \end{aligned}$$

□

7.3.0.3 The Pythagorean theorem This section recapitulates and extends our findings from the previous two sections. Let

$$\varphi(a) = (u - av, u - av) \tag{67}$$

$$= (u, u) + a^2 \cdot (v, v) - a \cdot \underbrace{[(u, v) + (v, u)]}_{=2\mathrm{Re}(u,v)} \tag{68}$$

The smallest distance is the minimum of (68). A necessary condition is

$$\varphi'(a) = 2a \cdot (v, v) - 2\mathrm{Re}(u, v) = 0$$

The minimum is

$$a_0 = \frac{\mathrm{Re}(u, v)}{(v, v)}.$$

Hence, we have found a minimal point a_0, which fulfills the condition $0 \leq \varphi(a_0) \leq \varphi(a)$. In detail:

$$(u, u) - a_0 \cdot 2\mathrm{Re}(u, v) + a_0^2 \cdot (v, v) \quad \leq \quad (u, u) + a^2 \cdot (v, v) - a^2 \cdot \mathrm{Re}(u, v).$$

Furthermore

$$\begin{aligned} 0 &\leq (u, u) - \frac{2\mathrm{Re}(u, v)\,\mathrm{Re}(u, v)}{(v, v)} + \frac{|\mathrm{Re}(u, v)|^2}{(v, v)^2}\,(v, v) \\ &= (u, u) - \frac{|\mathrm{Re}(u, v)|^2}{(v, v)}. \end{aligned}$$

It follows:

$$|\mathrm{Re}(u, v)| \leq (u, u) \cdot (v, v)$$

We now define the **projection** of a vector w onto a vector v:

$$P_{<v>}(w) = \frac{(w, v)}{(v, v)} \cdot v.$$

For an illustration, we refer to Figure 15.

7.3.0.4 Properties of the projection We investigate now the properties of the projection:

i) $P_{<v>}^2 = P_{<v>}$ (Idempotenz (germ.))

ii) $P_{<v>}^* = P_{<v>}$ (Self-adjointness)

The second property can be shown easily using the scalar product. It says that the projection operator can be shifted into the second argument. It follows that

$$(P_{<v>}(w), w') = (w, P_{<v>}(w')) \quad \Leftrightarrow \quad P_{<v>}^* = P_{<v>}$$

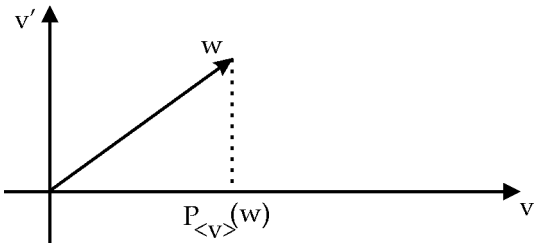


Figure 15: Projection of w on v.

Proof. To i)
Let $w' = P_{<v>}(w)$. We compute

$$P^2_{<v>}(w) = P_{<v>}(w') = P_{<v>}\left[\frac{(w,v)}{(v,v)} \cdot v\right] = \frac{(w,v)}{(v,v)} \underbrace{P_{<v>}(v)}_{=v} = P_{<v>}(w)$$

To ii)
On the one hand, we have

$$(P_{<v>}(w), w') = \frac{(w,v)}{(v,v)}(v, w').$$

Secondly,

$$(w, P_{<v>}(w')) = \left(w, \frac{(w',v)}{(v,v)} \cdot v\right) = \frac{(v,w') \cdot (w,v)}{(v,v)}$$

Comparing both sides yields the assertion. □

7.3.0.5 The projection theorem (the Pythagorean theorem) Using Figure 15, we can derive the Pythagorean theorem. It holds

$$\begin{aligned} ||w||^2 &= ||P_{<v>}(w) + (w - P_{<v>}(w))||^2 \\ &= ||P_{<v>}(w)||^2 + ||w - P_{<v>}(w)||^2 + (P_{<v>}(w), w - P_{<v>(w)}) + \overline{(\dots)} \\ &= ||P_{<v>}(w)||^2 + ||w - P_{<v>}(w)||^2 + (P_{<v>(w)}, w) - (P_{<v>}(w), P_{<v>}(w)) \end{aligned}$$

Furthermore, using $P^*_{<v>}P_{<v>} = P^2_{<v>} = P_{<v>}$, we obtain

$$\begin{aligned} ||w||^2 &= ||P_{<v>}(w)||^2 + ||w - P_{<v>}(w)||^2 + (P_{<v>(w)}, w) - (P^*_{<v>}P_{<v>}(w), w) \\ &= ||P_{<v>}(w)||^2 + ||w - P_{<v>}(w)||^2. \end{aligned}$$

Proposition 7.60 (Pythagorean theorem)**.** *We define:*

$$||w||^2 = ||P_{<v>}(w)||^2 + ||w - P_{<v>}(w)||^2$$

Specifically, it holds

$$||w||^2 \geq ||P_{<v>}(w)||^2 \tag{69}$$

The equality is obtained for $w = ||P_{<v>}(w)||$.

Remark 7.61. *We can write in terms of scalar products:*

$$(w,w) \geq \frac{|(w,v)|^2}{(v,v)^2} \cdot (v,v)$$

and can obtain backwards the Cauchy-Schwarz inequality:

$$(w,w)(v,v) \geq |(w,v)|^2.$$

7.4 Differentiation in Banach spaces

We discuss in this section how to differentiate in Banach spaces.

Definition 7.62 (Directional derivative in a Banach space)**.** *Let V and W be normed vector spaces and let $U \subset V$ be non-empty. Let $f : U \to W$ be a given mapping. If the limit*

$$f'(v)(h) := \lim_{\varepsilon \to 0} \frac{f(v+\varepsilon h) - f(v)}{\varepsilon} = \frac{d}{d\varepsilon} f(v+\varepsilon h)|_{\varepsilon=0}, \quad v \in U, h \in V$$

exists, then $f'(v)(h)$ is called the directional derivative of the mapping f at v in direction h. If the directional derivative exists for all $h \in V$, then f is called directionally differentiable at v.

Remark 7.63 (on the notation)**.** *Often, the direction h is denoted by δv in order to highlight that the direction is related with the variable v. This notation is useful, when several solution variables exist and several directional derivatives need to be computed.*

Remark 7.64. *The definition of the directional derivative in Banach spaces is in perfect agreement with the definition of derivatives in $\mathbb{R}$ at $x \in \mathbb{R}$ (see [85]):*

$$f'(x) := \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon) - f(x)}{\varepsilon},$$

and in $\mathbb{R}^n$ we have (see [86]):

$$f'(x)(h) := \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon h) - f(x)}{\varepsilon}.$$

*A function is called differentiable when all directional derivatives exist in the point $x \in \mathbb{R}^n$ (similar to the Gâteaux derivative). The derivatives in the directions $e_i, i = 1, \ldots, n$ of the standard basis are the well-known **partial derivatives**.*

Example 7.65. *Given $f(u) = u^3$, compute the directional derivative into $h \in V$. That is:*

$$f'(u)(h) = 3u^2 h.$$

Definition 7.66 (Gâteaux derivative)**.** *Let the assumptions hold as in Definition 7.62. A directional-differentiable mapping as defined in Definition 7.62, is called Gâteaux differentiable at $v \in U$, if there exists a linear continuous mapping $A \in L(U, W)$ such that*

$$f'(v)(h) = A(h)$$

holds true for all $h \in U$. Then, A is the Gâteaux derivative of f at v and we write $A = f'(v)$.

Remark 7.67. *The definition of the directional derivative immediately carries over to bilinear forms and for this reason the derivatives in Example 9.26 are well-defined. Therein, the semilinear form $a(u, z)$ is differentiated with respect to the first variable u into the direction $\phi \in V$.*

Remark 7.68. *The Gâteaux is computed with the help of directional deriatives and it holds $f'(v) \in L(U, W)$. If $W = \mathbb{R}$, then $f'(v) \in U^*$.*

Definition 7.69 (Fréchet derivative)**.** *A mapping $f : U \to W$ is Fréchet differentiable at $u \in U$ if there exists an operator $A \in L(U, W)$ and a mapping $r(u, \cdot) : U \to W$ such that for each $h \in U$ with $u + h \in U$, it holds:*

$$f(u+h) = f(u) + A(h) + r(u, h)$$

with

$$\frac{\|r(u,h)\|_W}{\|h\|_U} \to 0 \quad \text{for } \|h\|_U \to 0.$$

The operator $A(\cdot)$ is the Fréchet derivative of f at u and we write $A = f'(u)$.

Example 7.70. *Given $f(u) = u^3$, we show that the derivative is Fréchet:*

$$f(u+h) = (u+h)^3 = \underbrace{u^3}_{=f(u)} + \underbrace{3u^2h}_{=A(u)} + \underbrace{3uh^2 + h^3}_{=r(u,h)}$$

with $A(u) = f'(u)(h) = 3u^2h$ and clearly

$$\frac{\|r(u,h)\|}{\|h\|} \to 0.$$

Example 7.71. *The above bilinear form $a(u,\phi) = (\nabla u, \nabla\phi)$ is Fréchet differentiable in the first argument u (of course also in the second argument! But u is the variable we are interested in):*

$$a(u+h,\phi) = (\nabla(u+h), \nabla\phi) = \underbrace{(\nabla u, \nabla\phi)}_{=a(u,\phi)} + \underbrace{(\nabla h, \nabla\phi)}_{=a'_u(u,\phi)(h)}.$$

Here the remainder term is zero, i.e., $r(u,h) = 0$, because the bilinear form is linear in u. Thus, the Fréchet derivative of $a(u,\phi) = (\nabla u, \nabla\phi)$ is $a'_u(u,\phi)(h) = (\nabla h, \nabla\phi)$.

Example 7.72. *Another example is a functional of the form $J(u) = \int u^2\,dx$. Here:*

$$J(u+h) = \int (u+h)^2\,dx = \underbrace{\int u^2\,dx}_{J(u)} + \underbrace{\int 2uh\,dx}_{=A(h)} + \underbrace{\int h^2\,dx}_{=r(u,h)}.$$

That $J(u)$ is really Fréchet differentiable we need to check whether

$$\frac{\|r(u,h)\|_W}{\|h\|_U} \to 0 \quad \textit{for } \|h\|_U \to 0.$$

Here we have

$$\int h^2\,dx = \|h\|_V^2$$

and therefore:

$$\frac{\|h\|_V^2}{\|h\|_V} = \|h\|_V$$

For $h \to 0$ we clearly have $\|h\|_V \to 0$. Consequently the directional derivative of $J(u) = \int u^2\,dx$ is

$$J'(u)(h) = A(h) = \int_\Omega 2uh\,dx.$$

Example 7.73. *We present four further examples in which two variables couple. Such situations arise in Chapter 14. Now it depends whether both are unknowns simultaneously or whether they can be decoupled by assuming that one variable is given and only one is actually unknown. We consider*

1. *Solve u and v. Let $f(U) = u + v$ with $U = (u,v)$.*
2. *Solve u and v. Let $f(U) = uv$ with $U = (u,v)$.*
3. *Given v and solve for u. Let $f(U) = u + v$ with $U = (u,v)$.*
4. *Given v and solve for u. Let $f(U) = uv$ with $U = (u,v)$.*

The derivatives read:
$f'(U)(\delta U) = \delta u + \delta v$. *Because*

$$f'(U)(\delta U) = \lim_{\varepsilon\to 0} \frac{(u + \varepsilon\delta u + v + \varepsilon\delta v) - (u+v)}{\varepsilon} = \delta u + \delta v.$$

$f'(U)(\delta U) = v\delta u + u\delta v$. *Because*

$$f'(U)(\delta U) = \lim_{\varepsilon \to 0} \frac{(u + \varepsilon\delta u)(v + \varepsilon\delta v) - (uv)}{\varepsilon} = \lim_{\varepsilon \to 0} \frac{\varepsilon\delta uv + \varepsilon\delta vu + \varepsilon^2\delta u\delta v}{\varepsilon} = \delta uv + \delta vu.$$

$f'(U)(\delta u) = \delta u$. *Because*

$$f'(U)(\delta U) = \lim_{\varepsilon \to 0} \frac{(u + \varepsilon\delta u + v) - (u + v)}{\varepsilon} = \delta u.$$

$f'(U)(\delta u) = v\delta u$. *Because*

$$f'(U)(\delta U) = \lim_{\varepsilon \to 0} \frac{(u + \varepsilon\delta u)v - (uv)}{\varepsilon} = v\delta u.$$

8 Theory and finite elements (FEM) for elliptic problems

In this section, we discuss the finite element method (FEM). Our strategy is to introduce the method first for 1D problems in which deeper mathematical results with respect to functional analysis and Sobolev theory are not required for understanding. Later we also provide deeper mathematical results. The discretization parts are again complemented with the corresponding numerical analysis as well as algorithmic aspects and prototype numerical tests. As before, we augment from time to time with nonstandard formulations, such as for instance formulating a prototype coupled problem. Historically, the work by R. Courant in 1943 [37] is considered to be the first mathematical contribution to the finite element method. At the end of the 60's, engineers used finite elements and from then on, fruitful studies in both disciplines mathematics and engineering appeared.

8.1 Preliminaries

8.1.1 Construction of an exact solution (only possible for simple cases!)

In this section, we construct again an exact solution first. In principle this section is the same as Section 6.2.1 except that we derive a very explicit representation of the solution (but we could have done this in Section 6.2.1 as well). We consider again the model problem (D)[4]:

$$-u'' = f \quad \text{in } \Omega = (0,1), \tag{70}$$

$$u(0) = u(1) = 0. \tag{71}$$

Please make yourself clear that this is nothing else than expressing Formulation 4.28 in 1D. Thus, we deal with a second-order elliptic problem.

In 1D, we can derive the explicit solution of (70). First we have

$$u'(\tilde{x}) = -\int_0^{\tilde{x}} f(s)ds + C_1$$

where C_1 is a positive constant. Further integration yields

$$u(x) = \int_0^x u'(\tilde{x})\, d\tilde{x} = -\int_0^x \Big(\int_0^{\tilde{x}} f(s)\, ds\Big) d\tilde{x} + C_1 x + C_2$$

with another integration constant C_2. We have two unknown constants C_1, C_2 but also two given boundary conditions which allows us to determine C_1 and C_2.

$$0 = u(0) = -\int_0^x \Big(\int_0^{\tilde{x}} f(s)\, ds\Big) d\tilde{x} + C_1 \cdot 0 + C_2$$

yields $C_2 = 0$. For C_1, using $u(1) = 0$, we calculate:

$$C_1 = \int_0^1 \Big(\int_0^{\tilde{x}} f(s)\, ds\Big)\, d\tilde{x}.$$

Thus we obtain as final solution:

$$u(x) = -\int_0^x \Big(\int_0^{\tilde{x}} f(s)\, ds\Big) d\tilde{x} + x \left(\int_0^1 \Big(\int_0^{\tilde{x}} f(s)\, ds\Big)\, d\tilde{x}\right).$$

Thus, for a given right hand side f we obtain an explicit expression. For instance $f = 1$ yields:

$$u(x) = \frac{1}{2}(-x^2 + x).$$

It is trivial to double-check that this solution also satisfies the boundary conditions: $u(0) = u(1) = 0$. Furthermore, we see by differentiation that the original equation is obtained:

$$u'(x) = -x + \frac{1}{2} \quad \Rightarrow \quad -u''(x) = 1.$$

[4] (D) stands for differential problem. (M) stands for minimization problem. (V) stands for variational problem.

Exercise 5. *Let $f = -1$.*

- *Compute C_1;*
- *Compute $u(x)$;*
- *Check that $u(x)$ satisfies the boundary conditions;*
- *Check that $u(x)$ satisfies the PDE.*

8.1.2 Equivalent formulations

We first discuss that the solution of (70) (D) is also the solution of a minimization problem (M) and a variational problem (V). To formulate these (equivalent) problems, we first introduce the scalar product

$$(v, w) = \int_0^1 v(x)w(x)\,dx.$$

Furthermore we introduce the linear space

$$V := \{v|\ v \in C[0,1], v' \text{ is piecewise continuous and bounded on } [0,1], v(0) = v(1) = 0\}. \tag{72}$$

We also introduce the linear functional $F : V \to \mathbb{R}$ such that

$$F(v) = \frac{1}{2}(v', v') - (f, v).$$

Recall from Numerik 1 [114][Section 7.5]

$$Q(x) = \frac{1}{2}(Ax, x)_2 - (b, x)_2.$$

We state

Definition 8.1 (Four equivalent relationships)**.** *We deal with four (equivalent) problems:*

(D) Find $u \in C^2$ such that $-u'' = f$ with $u(0) = u(1) = 0$; (strong form)

(M) Find $u \in V$ such that $F(u) \leq F(v)$ for all $v \in V$; (energy minimization)

(V) Find $u \in V$ such that $(u', v') = (f, v)$ for all $v \in V$; (weak form).

Moreover, the conservation relation (see Section 4.1.3) is equivalent to these three problems:

(C) Find $u \in C^2$ such that $(-u'', 1) = (f, 1)$ with $u(0) = u(1) = 0$ (volume conservation law).

*Actually, (C) is most often the starting point from a physical derivation using **volume conservation principles** such as mass, momentum (angular momentum), and energy. In physics, the quantity $F(v)$ stands for the **total potential energy** of the underlying model; see also Section 9.6.2. Moreover, the first term in $F(v)$ denotes the internal elastic energy and (f, v) the load potential. Therefore, formulation (M) corresponds to the fundamental **principle of minimal potential energy** and the variational problem (V) to the **principle of virtual work** (e.g., [34]). The proofs of their equivalence will be provided in the following.*

Remark 8.2 (Euler-Lagrange equations)**.** *When (V) is derived from (M), we also say that (V) is the **Euler-Lagrange equation** related to (M).*

Remark 8.3 (Numerical schemes inspired by these four equivalent relations)**.** *The previous four relations inspire different discretization schemes:*

(D) $\rightarrow$ Finite differences; Chapter 6.

(M) $\rightarrow$ Optimization algorithms

(V) $\rightarrow$ *Galerkin finite elements; this chapter*

(C) $\rightarrow$ *Finite volume schemes*

In the following we show that the four problems $(D), (M), (V), (C)$ are equivalent.

Proposition 8.4. *It holds*

$$(D) \leftrightarrow (C).$$

Proof. First, $(D) \rightarrow (C)$ is trivial. The other way around, i.e. $(C) \rightarrow (D)$ has been discussed in Section 4.1.3, namely in (C) the integrand must be sufficiently regular and the integral relation must hold for arbitrary subsets U of Ω. □

As next relationship, we have:

Proposition 8.5. *It holds*

$$(D) \rightarrow (V).$$

Proof. We multiply $u'' = f$ with an arbitrary function ϕ (a so-called **test function**) from the space V defined in (72). Then we integrate over the interval $\Omega = (0, 1)$ yielding

$$-u'' = f \tag{73}$$

$$\Rightarrow -\int_\Omega u''\phi \, dx = \int_\Omega f\phi \, dx \tag{74}$$

$$\Rightarrow \int_\Omega u'\phi' \, dx - u'(1)\phi(1) + u'(0)\phi(0) = \int_\Omega f\phi \, dx \tag{75}$$

$$\Rightarrow \int_\Omega u'\phi' \, dx = \int_\Omega f\phi \, dx \quad \forall \phi \in V. \tag{76}$$

In the second last term, we used integration by parts.

The boundary terms vanish because $\phi \in V$. This shows that

$$\int_\Omega u'\phi' \, dx = \int_\Omega f\phi \, dx$$

is a solution of (V). □

Proposition 8.6. *It holds*

$$(V) \leftrightarrow (M).$$

Proof. We first assume that u is a solution to (V). Let $\phi \in V$ and set $w = \phi - u$ such that $\phi = u + w$ and $w \in V$. We obtain

$$\begin{aligned} F(\phi) = F(u + w) &= \frac{1}{2}(u' + w', u' + w') - (f, u + w) \\ &= \frac{1}{2}(u', u') - (f, u) + (u', w') - (f, w) + \frac{1}{2}(w', w') \geq F(u) \end{aligned}$$

We use now the fact that (V) holds true, namely

$$(u', w') - (f, w) = 0,$$

and also that $(w', w') \geq 0$. Thus, we have shown that u is a solution to (M). We show now that $(M) \rightarrow (V)$ holds true as well. For any $\phi \in V$ and $\varepsilon \in \mathbb{R}$ we have

$$F(u) \leq F(u + \varepsilon\phi),$$

because $u + \varepsilon\phi \in V$. We differentiate with respect to ε and show that (V) is a first order necessary condition to (M) with a minimum at $\varepsilon = 0$. To do so, we define

$$g(\varepsilon) := F(u + \varepsilon\phi) = \frac{1}{2}(u', u') + \varepsilon(u', \phi') + \frac{\varepsilon^2}{2}(\phi', \phi') - (f, u) - \varepsilon(f, \phi).$$

Thus

$$g'(\varepsilon) = (u', \phi') + \varepsilon(\phi', \phi') - (f, \phi).$$

A minimum is obtained for $\varepsilon = 0$. Consequently,

$$g'(0) = 0.$$

In detail:

$$(u', \phi') - (f, \phi) = 0,$$

which is nothing else than the solution of (V). □

Proposition 8.7. *The solution u to (V) is unique.*

Proof. Assume that u_1 and u_2 are solutions to (V) with

$$(u_1', \phi') = (f, \phi) \quad \forall \phi \in V,$$
$$(u_2', \phi') = (f, \phi) \quad \forall \phi \in V.$$

We subtract these solutions and obtain:

$$(u_1' - u_2', \phi') = 0.$$

We choose as test function $\phi' = u_1' - u_2'$ and obtain

$$(u_1' - u_2', u_1' - u_2') = 0.$$

Thus:

$$u_1'(x) - u_2'(x) = (u_1 - u_2)'(x) = 0.$$

Since the difference of the derivatives is zero, this means that the difference of the functions themselves is constant:

$$(u_1 - u_2)(x) = const$$

Using the boundary conditions $u(0) = u(1) = 0$, yields

$$(u_1 - u_2)(x) = 0.$$

Thus $u_1 = u_2$. □

Remark 8.8. *The last statements are related to the definiteness of the norm. It holds:*

$$\|u\|_{L^2}^2 = \int_\Omega u^2 \, dx.$$

Thus the results follow directly because

$$\|u\| = 0 \quad \Leftrightarrow \quad u = 0.$$

Proposition 8.9. *It holds*

$$(V) \to (D).$$

Proof. We assume that u is a solution to (V), i.e.,

$$(u', \phi') = (f, \phi) \quad \forall \phi \in V.$$

If we assume sufficient regularity of u (in particular $u \in C^2$), then u'' exists and we can integrate backwards. Moreover, we use that $\phi(0) = \phi(1) = 0$ since $\phi \in V$. Then:

$$(-u'' - f, \phi) = 0 \quad \forall \phi \in V.$$

Since we assumed sufficient regularity for u'' and f the difference is continuous. We can now apply the fundamental principle (see Proposition 8.11):

$$w \in C(\Omega) \quad \Rightarrow \quad \int_\Omega w\phi \, dx = 0 \quad \Rightarrow \quad w \equiv 0.$$

We proof this result later. Before, we obtain

$$(-u'' - f, \phi) = 0 \quad \Rightarrow \quad -u'' - f = 0.$$

This yields the PDE. The boundary conditions are recovered because $u \in V$ and therein, we have $u(0) = u(1) = 0$. Therefore, we obtain:

$$\begin{aligned} -u'' &= f \\ u(0) = (1) &= 0. \end{aligned}$$

Since we know that $(D) \to (V)$ holds true, u has the assumed regularity properties and we have shown the equivalence. □

It remains to state and proof the fundamental lemma of calculus of variations. To this end, we introduce

Definition 8.10 (Continuous functions with compact support)**.** *Let $\Omega \subset \mathbb{R}^n$ be an open domain.*

- *The set of continuous functions from Ω to $\mathbb{R}$ with **compact support** is denoted by $C_c(\Omega)$. Such functions vanish on the boundary.*
- *The set of **smooth functions** (infinitely continuously differentiable) with compact support is denoted by $C_c^\infty(\Omega)$. Often, functions $\varphi \in C_c^\infty(\Omega)$ are so-called **test functions**.*

Proposition 8.11 (Fundamental lemma of calculus of variations)**.** *Let $\Omega = [a, b]$ be a compact interval and let $w \in C(\Omega)$. Let $\phi \in C^\infty$ with $\phi(a) = \phi(b) = 0$, i.e., $\phi \in C_c^\infty(\Omega)$. If for all ϕ it holds*

$$\int_\Omega w(x)\phi(x)\,dx = 0, \tag{77}$$

then, $w \equiv 0$ in Ω.

Proof. We perform an indirect proof. We suppose that there exist a point $x_0 \in \Omega$ with $w(x_0) \neq 0$. Without loss of generality, we can assume $w(x_0) > 0$. Since w is continuous, there exists a small (open) neighborhood $\omega \subset \Omega$ with $w(x) > 0$ for all $x \in \omega$ and outside $w \equiv 0$ in $\Omega \setminus \omega$. Let ϕ now be a positive test function (recall that ϕ can be arbitrary, specifically positive if we wish) in Ω and thus also in ω. Since $w(x)$ and $\phi(x)$ are continuous the product is also $w(x)\phi(x) \in C(\Omega)$. But now $w(x)\phi(x) > 0$ in ω and $w(x)\phi(x) = 0$ in $\Omega \setminus \omega$. By using next the hypothesis (77), we have

$$0 = \int_\Omega w(x)\phi(x)\,dx = \int_\omega w(x)\phi(x)\,dx > 0.$$

But this is a contradiction. Thus $w(x) = 0$ for all in $x \in \omega$. Extending this result to all open neighborhoods in Ω we arrive at the final result. The same procedure can be applied to $w(x_0) < 0$ and consequently, $w \equiv 0$ in Ω. □

8.2 Derivation of a weak form

We continue to consider the variational form in this section and provide more definitions and notation. We recall the key idea, which is a **two-step procedure**:

- Step 1: Design a function space V that also includes the correct boundary conditions.
- Step 2: Multiply the strong form (D) with a test function from V and integrate.

The second operation 'weakens' the derivative information (therefore the name **weak form** because rather evaluating 2nd order derivatives, we only need to evaluate a 1st order derivative on the trial function and another 1st order derivative on the test function.

For Poisson with homogeneous Dirichlet conditions, we then obtain for Step 1: Take space

$$V := \{v |\ v \in C[0,1], v' \text{ is pc. cont. and bound. on } [0,1], v(0) = v(1) = 0\}$$

from before. We now address Step 2:

$$-u'' = f \tag{78}$$

$$\Rightarrow -\int_\Omega u''\phi\,dx = \int_\Omega f\phi\,dx \tag{79}$$

$$\Rightarrow \int_\Omega u'\phi'\,dx - \int_{\partial\Omega} \partial_n u\phi\,ds = \int_\Omega f\phi\,dx \tag{80}$$

$$\Rightarrow \int_\Omega u'\phi'\,dx = \int_\Omega f\phi\,dx. \tag{81}$$

To summarize we have:

$$\int_\Omega u'\phi'\,dx = \int_\Omega f\phi\,dx \tag{82}$$

A common short-hand notation in mathematics is to use parentheses for L^2 scalar products: $\int_\Omega ab\,dx =: (a,b)$:

$$(u',\phi') = (f,\phi) \tag{83}$$

A mathematically-correct statement is:

Formulation 8.12. *Find $u \in V$ such that*

$$(u',\phi') = (f,\phi) \quad \forall\phi \in V. \tag{84}$$

8.3 Notation: writing the bilinear and linear forms in abstract notation

In the following, a very common notation is introduced. First, we recall concepts known from linear algebra:

Definition 8.13 (Linear form and bilinear form)**.** *If V is a linear space, l is a **linear form** on V if $l : V \to \mathbb{R}$ or in other words $l(v) \in \mathbb{R}$ for $v \in V$. Moreover, l is linear:*

$$l(av + bw) = al(v) + bl(w) \quad \forall a,b \in R, \quad \forall v,w \in V.$$

*A problem is a **bilinear form** [5] on $V \times V$ if $a : V \times V \to \mathbb{R}$ and $a(v,w) \in \mathbb{R}$ for $v,w \in V$ and $a(v,w)$ is linear in each argument, i.e., for $\alpha,\beta \in R$ and $u,v,w \in V$, it holds*

$$a(u,\alpha v + \beta w) = \alpha a(u,v) + \beta a(u,w),$$
$$a(\alpha u + \beta v, w) = \alpha a(u,w) + \beta a(v,w).$$

*A bilinear form is said to be **symmetric** if*

$$a(u,v) = a(v,u).$$

*A symmetric bilinear form $a(\cdot,\cdot)$ on $V \times V$ defines a **scalar product** on V if*

$$a(v,v) > 0 \quad \forall v \in V, v \neq 0.$$

*The associated **norm** is denoted by $\|\cdot\|_a$ and defined by*

$$\|v\|_a := \sqrt{a(v,v)} \quad \text{for } v \in V.$$

Definition 8.14 (Frobenius scalar product)**.** *For vector-valued functions in second-order problems, we need to work with a scalar product defined for matrices because their gradients are matrices. Here the natural form is the Frobenius scalar product, which is defined as:*

$$< A,B >_F = A : B = \sum_i \sum_j a_{ij}b_{ij}$$

The Frobenius scalar product induces the Frobenius norm:

$$\|A\|_F = \sqrt{< A,B >}.$$

[5] For nonlinear problems we deal with a semi-linear form, which is only linear in one argument, i.e., the solution variable and where the test function enters linearly; see Chapter 13.

Remark 8.15. *For vector-valued PDEs (such as elasticity, Stokes or Navier-Stokes), we formulate in compact form:*

$$a(u, \phi) = l(\phi)$$

with some $l(\phi)$ *and*

$$a(u, \phi) = \int_\Omega \nabla u : \nabla \phi \, dx$$

where $\nabla u : \nabla \phi$ *is then evaluated in terms of the Frobenius scalar product.*

Definition 8.16 (Bilinear and linear forms of the continuous Poisson problem)**.** *For the Poisson problem, we can define:*

$$a(u, \phi) = (u', \phi'),$$
$$l(\phi) = (f, \phi).$$

We shall state the variational form of Poisson's problem in terms of $a(\cdot, \cdot)$ and $l(\cdot)$:

Formulation 8.17 (Variational Poisson problem on the continuous level)**.** *Find* $u \in V$ *such that*

$$a(u, \phi) = l(\phi) \quad \forall \phi \in V, \tag{85}$$

where $a(\cdot, \cdot)$ *and* $l(\cdot)$ *are defined in Definition 8.16. The unknown function* u *is called the* ***trial*** *or* ***(ansatz)*** *function whereas* ϕ *is the so-called* ***test function****.*

8.4 Finite elements in 1D

In the following we want to concentrate how to compute a discrete solution. As in the previous chapter, this, in principle, allows us to address even more complicated situations and also higher spatial dimensions. The principle of the FEM is rather simple:

1. Introduce a mesh $\mathcal{T}_h := \bigcup K_i$ (where K_i denote the single mesh elements) of the given domain $\Omega = (0, 1)$ with mesh size (diameter/length) parameter h
2. Define on each mesh element $K_i := [x_i, x_{i+1}], i = 0, \ldots, n$ polynomials for trial and test functions. These polynomials must form a basis in a space V_h and they should reflect certain conditions on the mesh edges;
3. Use the variational (or weak) form of the given problem and derive a discrete version. Then, plug-in a linear combination of basis functions from V_h;
4. Evaluate the arising integrals;
5. Collect all contributions on all K_i leading to a linear equation system $AU = B$;
6. Solve this linear equation system; the solution vector $U = (u_0, \ldots, u_n)^T$ contains the discrete solution at the nodal points $x_0, \ldots, x_n$ (for Lagrange FEM);
7. Verify the correctness of the solution U (experimentally, numerical/computational analysis, analytical results).

8.4.1 The mesh

Let us start with the mesh. We introduce nodal points and divide $\Omega = (0, 1)$ into

$$x_0 = 0 < x_1 < x_2 < \ldots < x_n < x_{n+1} = 1.$$

In particular, we can work with a uniform mesh in which all nodal points have equidistant distance:

$$x_j = jh, \quad h = \frac{1}{n+1}, \quad 0 \le j \le n+1, \quad h = x_{j+1} - x_j.$$

Remark 8.18. *An important research topic is to organize the points* x_j *in certain non-uniform ways (keyword is adaptivity!) in order to reduce the discrete error (see Section 8.13).*

8.4.2 Linear finite elements

In the following we denote P_k the space that contains all polynomials up to order k with coefficients in $\mathbb{R}$:

Definition 8.19.

$$P_k := \{\sum_{i=0}^{k} a_i x^i |\ a_i \in \mathbb{R}\}.$$

In particular we will work with the space of linear polynomials

$$P_1 := \{a_0 + a_1 x |\ a_0, a_1 \in \mathbb{R}\}.$$

A finite element is now a function localized to an element $K_i \in \mathcal{T}_h$ and uniquely defined by the values in the nodal points x_i, x_{i+1}.

We then define the space:

$$V_h^{(1)} = V_h := \{v \in C[0,1] |\ v|_{K_i} \in P_1, K_i := [x_i, x_{i+1}], 0 \leq i \leq n, v(0) = v(1) = 0\}.$$

The boundary conditions are build into the space through $v(0) = v(1) = 0$. This is an important concept that Dirichlet boundary conditions will not appear explicitly later, but are contained in the function spaces.

All functions inside V_h are so called **shape functions** and can be represented by so-called **hat functions**. Hat functions are specifically linear functions on each element K_i. Attaching them yields a hat in the geometrical sense.

For $j = 1, \ldots, n$ we define:

$$\phi_j(x) = \begin{cases} 0 & \text{if } x \notin [x_{j-1}, x_{j+1}] \\ \frac{x - x_{j-1}}{x_j - x_{j-1}} & \text{if } x \in [x_{j-1}, x_j] \\ \frac{x_{j+1} - x}{x_{j+1} - x_j} & \text{if } x \in [x_j, x_{j+1}] \end{cases} \tag{86}$$

with the property

$$\phi_j(x_i) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}. \tag{87}$$

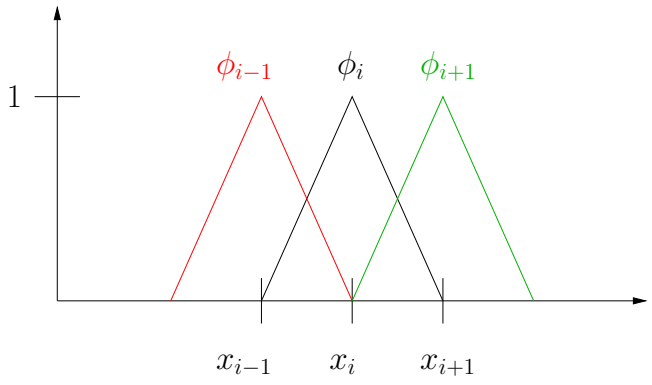


Figure 16: Hat functions.

For a uniform step size $h = x_j - x_{j-1} = x_{j+1} - x_j$ we obtain

$$\phi_j(x) = \begin{cases} 0 & \text{if } x \notin [x_{j-1}, x_{j+1}] \\ \frac{x - x_{j-1}}{h} & \text{if } x \in [x_{j-1}, x_j] \\ \frac{x_{j+1} - x}{h} & \text{if } x \in [x_j, x_{j+1}] \end{cases}$$

and for its derivative:

$$\phi_j'(x) = \begin{cases} 0 & \text{if } x \notin [x_{j-1}, x_{j+1}] \\ +\frac{1}{h} & \text{if } x \in [x_{j-1}, x_j] \\ -\frac{1}{h} & \text{if } x \in [x_j, x_{j+1}] \end{cases}$$

Lemma 8.20. *The space V_h is a subspace of $V := C[0,1]$ and has dimension n (because we deal with n basis functions). Thus the such constructed finite element method is a **conforming** method. Furthermore, for each function $v_h \in V_h$ we have a unique representation:*

$$v_h(x) = \sum_{j=1}^{n} v_{h,j}\phi_j(x) \quad \forall x \in [0,1], \quad v_{h,j} \in \mathbb{R}.$$

Proof. Sketch: The unique representation is clear, because in the nodal points it holds $\phi_j(x_i) = \delta_{ij}$, where δ_{ij} is the Kronecker symbol with $\delta_{ij} = 1$ for $i = j$ and 0 otherwise. □

The function $v_h(x)$ connects the discrete values $v_{h,j} \in \mathbb{R}$ and in particular the values between two support points x_j and x_{j+1} can be evaluated.

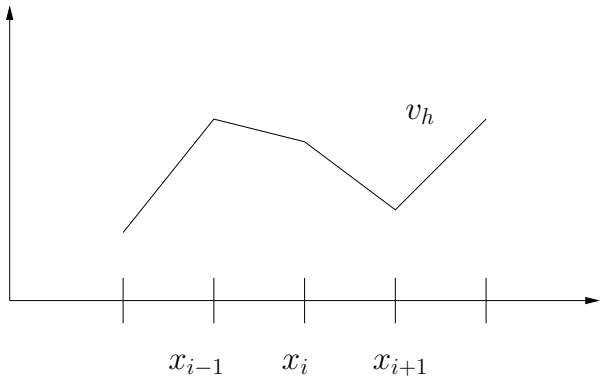


Figure 17: The function $v_h \in V_h$.

Remark 8.21. *The finite element method introduced above is a Lagrange method, since the basis functions ϕ_j are defined only through its values at the nodal points without using derivative information (which would result in Hermite polynomials).*

8.4.3 The process to construct the specific form of the shape functions

In the previous construction, we have hidden the process how to find the specific form of $\phi_j(x)$. For 1D it is more or less clear and we would accept the $\phi_j(x)$ really has the form as it has in (86) . In $\mathbb{R}^n$ this task is a bit of work. To understand this procedure, we explain the process in detail. Here we first address the defining properties of a finite element (see also Section 8.4.7):

- Intervals $[x_i, x_{i+1}]$;
- A linear polynomial $\phi(x) = a_0 + a_1 x$;
- Nodal values at x_i and x_{i+1} (the so-called degrees of freedom).

Later only these properties are stated (see also the general literature in Section 1) and one has to construct the specific $\phi(x)$ such as for example in (86). Thus, the main task consists in finding the unknown coefficients

a_0 and a_1 of the shape function. The key property is (87) (also valid in $\mathbb{R}^n$ in order to have a small support) and therefore we obtain:

$$\phi_j(x_j) = a_0 + a_1 x_j = 1,$$
$$\phi_j(x_i) = a_0 + a_1 x_i = 0.$$

To determine a_0 and a_1 we have to solve a small linear equation system:

$$\begin{pmatrix} 1 & x_j \\ 1 & x_i \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

We obtain

$$a_1 = -\frac{1}{x_i - x_j}$$

and

$$a_0 = \frac{x_i}{x_i - x_j}.$$

Then:

$$\phi_j(x) = a_0 + a_1 x = \frac{x_i - x}{x_i - x_j}.$$

At this stage we have now to distinguish whether $x_j := x_{i-1}$ or $x_j := x_{i+1}$ or $|i - j| > 1$ yielding the three cases in (86).

Remark 8.22. *Of course, for higher-order polynomials and higher-order problems in $\mathbb{R}^n$, the matrix system to determining the coefficients becomes larger. However, in all state-of-the-art FEM software packages, the shape functions are already implemented.*

Remark 8.23. *A very practical and detailed derivation of finite elements in different dimensions can be found in [120].*

8.4.4 The discrete weak form and employing linear combination for $u_h \in V_h$

We have set-up a mesh and local polynomial functions with a unique representation, all these developments result in a 'finite element'. Now, we use the variational formulation and derive the discrete counterpart:

Formulation 8.24. *Find $u_h \in V_h$ such that*

$$(u_h', \phi_h') = (f, \phi_h) \quad \forall \phi_h \in V_h. \tag{88}$$

Or in the previously introduced compact form:

Formulation 8.25 (Variational Poisson problem on the discrete level)**.** *Find $u_h \in V_h$ such that*

$$a(u_h, \phi_h) = l(\phi_h) \quad \forall \phi_h \in V_h, \tag{89}$$

where $a(\cdot, \cdot)$ and $l(\cdot)$ are defined in Definition 8.16 by adding h as subindex for u and ϕ.

Remark 8.26 (Galerkin method)**.** *The process going from V to V_h using the variational formulation is called **Galerkin method**. Here, it is not necessary that the bilinear form is symmetric. As further information: not only is Galerkin's method a numerical procedure, but it is also used in analysis when establishing existence of the continuous problem. Here, one starts with a finite dimensional subspace and constructs a sequence of finite dimensional subspaces $V_h \subset V$ (namely passing with $h \to 0$; that is to say: we add more and more basis functions such that $dim(V_h) \to \infty$). The idea of numerics is the same: finally we are interested in small h such that we obtain a discrete solution with sufficient accuracy.*

Remark 8.27 (Ritz method)**.** *If we discretize the minimization problem (M), the above process is called **Ritz method**. In particular, the bilinear form of the variational problem is symmetric.*

Remark 8.28 (Ritz-Galerkin method)**.** *For general bilinear forms (i.e., not necessarily symmetric) the discretization procedure is called* ***Ritz-Galerkin method****.*

Remark 8.29 (Petrov-Galerkin method)**.** *In a* ***Petrov-Galerkin method*** *the trial and test spaces can be different.*

In order to proceed we can express $u_h \in V_h$ with the help of the basis functions ϕ_j in $V_h := \{\phi_1, \dots, \phi_n\}$, thus:

$$u_h = \sum_{j=1}^{n} u_j \phi_j(x), \quad u_j \in \mathbb{R}.$$

Since (88) holds for all $\phi_i \in V_h$ for $1 \le i \le n$, it holds in particular for each i:

$$(u_h', \phi_i') = (f, \phi_i) \quad \text{for } 1 \le i \le n. \tag{90}$$

We now insert the representation for u_h in (90), yielding the **Galerkin equations**:

$$\sum_{j=1}^{n} u_j (\phi_j', \phi_i') = (f, \phi_i) \quad \text{for } 1 \le i \le n. \tag{91}$$

We have now extracted the coefficient vector of u_h and only the **shape functions** ϕ_j and ϕ_i (i.e., their derivatives of course) remain in the integral.

This yields a linear equation system of the form

$$AU = B$$

where

$$U = (u_j)_{1 \le j \le n} \in \mathbb{R}^n, \tag{92}$$

$$B = \big((f, \phi_i)\big)_{1 \le i \le n} \in \mathbb{R}^n, \tag{93}$$

$$A = (a_{ij})_{i,j=1}^{n} = \Big((\phi_j', \phi_i')\Big)_{1 \le j,i \le n} \in \mathbb{R}^{n \times n}. \tag{94}$$

Thus the final solution vector is U which contains the values u_j at the nodal points x_j of the mesh. Here we remark that x_0 and x_{n+1} are not solved in the above system and are determined by the boundary conditions $u(x_0) = u(0) = 0$ and $u(x_{n+1}) = u(1) = 0$.

Remark 8.30 (Regularity of A)**.** *It remains the question whether A is regular such that A^{-1} exists. We show this rigorously using Assumption No. 3 in Definition 8.71 in Section 8.12.3.*

Remark 8.31. *We discuss in Section 8.5.3, why $(a_{ij})_{i,j=1}^{n} = (\phi_j', \phi_i')$ holds and not $(a_{ij})_{i,j=1}^{n} = (\phi_i', \phi_j')$ in general.*

8.4.5 Evaluation of the integrals

It remains to determine the specific entries of the system matrix (also called stiffness matrix) A and the right hand side vector B. Since the basis functions have only little support on two neighboring elements (in fact that is one of the key features of the FEM) the resulting matrix A is sparse, i.e., it contains only a few number of entries that are not equal to zero.

Let us now evaluate the integrals that form the entries a_{ij} of the matrix $A = (a_{ij})_{1 \le i,j \le n}$. For the diagonal elements we calculate:

$$a_{ii} = \int_\Omega \varphi_i'(x)\varphi_i'(x)\,dx = \int_{x_{i-1}}^{x_{i+1}} \varphi_i'(x)\varphi_i'(x)\,dx \tag{95}$$

$$= \int_{x_{i-1}}^{x_i} \frac{1}{h^2}\,dx + \int_{x_i}^{x_{i+1}} \Big(-\frac{1}{h}\Big)^2\,dx \tag{96}$$

$$= \frac{h}{h^2} + \frac{h}{h^2} \tag{97}$$

$$= \frac{2}{h}. \tag{98}$$

For the right off-diagonal we have:

$$a_{i,i+1} = \int_\Omega \varphi'_{i+1}(x)\varphi'_i(x)\,dx = \int_{x_i}^{x_{i+1}} \frac{1}{h}\cdot\left(-\frac{1}{h}\right)dx = -\frac{1}{h}. \tag{99}$$

It is trivial to see that $a_{i,i+1} = a_{i-1,i}$.

In compact form we summarize:

$$a_{ij} = \int_\Omega \phi'_j(x)\phi'_i(x)\,dx = \begin{cases} -\frac{1}{x_{j+1}-x_j} & \text{if } j = i-1 \\ \frac{1}{x_j - x_{j-1}} + \frac{1}{x_{j+1}-x_j} & \text{if } j = i \\ -\frac{1}{x_j - x_{j-1}} & \text{if } j = i+1 \\ 0 & \text{otherwise} \end{cases}. \tag{100}$$

For a uniform mesh (as we assume in this section) we can simplify the previous calculation since we know that $h = h_j = x_{j+1} - x_j$:

$$a_{ij} = \int_\Omega \phi'_j(x)\phi'_i(x)\,dx = \begin{cases} -\frac{1}{h} & \text{if } j = i-1 \\ \frac{2}{h} & \text{if } j = i \\ -\frac{1}{h} & \text{if } j = i+1 \\ 0 & \text{otherwise} \end{cases}. \tag{101}$$

The resulting matrix A for the 'inner' points $x_1, \ldots, x_n$ reads then:

$$A = h^{-1}\begin{pmatrix} 2 & -1 & & & 0 \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ 0 & & & -1 & 2 \end{pmatrix} \in \mathbb{R}^{n\times n}.$$

Remark 8.32. *Since the boundary values at $x_0 = 0$ and $x_{n+1} = 1$ are known to be $u(0) = u(1) = 0$, they are not assembled in the matrix A. We could have considered all support points $x_0, x_1, \ldots, x_n, x_{n+1}$ we would have obtained:*

$$A = h^{-1}\begin{pmatrix} h & 0 & & & 0 \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ 0 & & & 0 & h \end{pmatrix} \in \mathbb{R}^{(n+2)\times(n+2)}.$$

Here the entries $a_{00} = a_{n+1,n+1} = 1$ (and not 2) because at the boundary only the half of the two corresponding test functions do exist. Furthermore, working with this matrix A in the solution process below, we have to modify the entries $a_{0,1} = a_{1,0} = a_{n,n+1} = a_{n+1,n}$ from the value -1 to 0 such that $u_{h,0} = u_{h,n+1} = 0$ can follow.

It remains to evaluate the integrals of the right hand side vector b. Here the main difficulty is that the given right hand side f may be complicated. Of course it always holds for the entries of b:

$$b_i = \int_\Omega f(x)\phi_i(x)\,dx \quad \text{for } 1 \le i \le n.$$

The right hand side now depends explicitly on the values for f. Let us assume a constant $f = 1$ (thus the original problem would be $-u'' = 1$), then:

$$b_i = \int_\Omega 1 \cdot \phi_i(x)\, dx = 1 \cdot \int_\Omega \phi_i(x)\, dx = 1 \cdot h \quad \text{for } 1 \leq i \leq n.$$

Namely

$$B = (h, \ldots, h)^T.$$

For non-uniform step sizes we obtain:

$$\frac{1}{h_i} \int_{K_i} \phi_i(x)\, dx, \quad \text{and} \quad \frac{1}{h_{i+1}} \int_{K_{i+i}} \phi_i(x)\, dx$$

with $h_i = x_i - x_{i-1}, h_{i+1} = x_{i+1} - x_i$ and $K_i = [x_{i-1}, x_i]$ and $K_{i+1} = [x_i, x_{i+1}]$.

Exercise 6. *Derive the system matrix A by assuming non-uniform step sizes $h_j = x_j - x_{j+1}$.*

Exercise 7. *Derive the system matrix A for a P_2 discretization, namely working with quadratic basis functions. Sketch how the basis functions look like.*

8.4.6 More comments on how to build-in boundary conditions

We briefly elaborate a little bit more on boundary conditions. If they are of Neumann type, they can be immediately assembled as previously shown. On the other hand, Dirichlet conditions must be carefully applied to the matrix. Usually the entries are locally modified as previously mentioned. In higher dimensions, this is more work, and constraint matrix can be used. An alternative to the strong prescription of Dirichlet conditions can be a weak formulation. The latter is as follows for instance:

Formulation 8.33 (Weak prescription of boundary conditions)**.** *Let Ω be a domain and Γ its boundary (sufficiently smooth). Let $V := H^1$. Let g be the boundary function and $\gamma > 0$ a penalization parameter. Find $u \in V$ such that*

$$a(u, \phi) + \gamma \int_\Gamma (u - g)\phi\, ds = l(\phi) \quad \forall \phi \in V.$$

8.4.7 Definition of a finite element

We briefly summarize the key ingredients that define a **finite element**. A finite element is a triple (K, P_K, Σ) where

- K is an element, i.e., a geometric object (in 1D an interval);
- $P_k(K)$ is a finite dimensional linear space of polynomials defined on K;
- Σ, not introduced so far, is a set of degrees of freedom (DoF), e.g., the values of the polynomial at the vertices of K.

These three ingredients yield a uniquely determined polynomial on an element K.

8.4.8 Properties of the system matrix A

We first notice:

Remark 8.34. *The system matrix A has a factor $\frac{1}{h}$ whereas in the corresponding matrix A, using a finite difference scheme, we have a factor $\frac{1}{h^2}$. The reason is that we integrate the weak form using finite elements and one h is hidden in the right hand side vector b. Division by h we yield the same system for finite elements as for finite differences.*

Next, we show that A is symmetric and positive definite. It holds

$$(\phi_i', \phi_j') = (\phi_j', \phi_i').$$

Using the representation

$$u(x) = \sum_{j=1}^{n} u_{j,h}\phi_{j,h}(x),$$

we obtain

$$\sum_{i,j=1}^{n} u_i(\phi_i', \phi_j')u_j = (\sum_{i,j=1}^{n} u_i\phi_i', \sum_{i,j=1}^{n} u_j\phi_j') = (u', u') \geq 0.$$

We have

$$(u', u') = 0$$

only if $u' \equiv 0$ since $u(0) = 0$ only for $u \equiv 0$ or $u_{j,h} = 0$ for $j = 1, \ldots, n$. We recall that a symmetric matrix $B \in \mathbb{R}^{n\times n}$ is said to be positive definite if

$$\xi \cdot B\xi = \sum_{i,j=1}^{n} \xi_i b_{ij} \xi_j > 0 \quad \forall \xi \in \mathbb{R}^n, \xi \neq 0.$$

Finally, it holds:

Proposition 8.35. *A symmetric positive matrix B is positive definite if and only if the eigenvalues of B are strictly positive. Furthermore, a positive definite matrix is regular and consequently, the linear system to which B is associated with has a unique solution.*

Proof. Results from linear algebra. □

8.4.9 Numerical test: 1D Poisson

We compute the same test as in Section 6.7, but now with finite elements. Since we can explicitly derive all entries of the system matrix A as in the finite difference method, everything would result in the same operations as in Section 6.7. Rather we adopt deal.II [6, 9] an open source C++ finite element code to carry out this computation. Even that in such packages many things are hidden and performed in the background, deal.II leaves enough room to **see** the important points:

- Creating a domain Ω and decomposition of this domain into elements;
- Setting up the vectors u, b and the matrix A with their correct length;
- Assembling (not yet solving!) the system $Au = b$. We compute locally on each mesh element the matrix entries and right hand side entries on a master cell and insert the respective values at their global places in the matrix A;
- In the assembly, the integrals are evaluated using numerical quadrature in order to allow for more general expressions when for instance the material parameter a is not constant to 1 or when higher-order polynomials need to be evaluated;
- Solution of the linear system $Au = b$ (the implementation of the specific method is hidden though;
- Postprocessing of the solution: here writing the solution data $U_j, 1 \leq j \leq n$ into a file that can be read from a graphic visualization program such as gnuplot, paraview, visit.

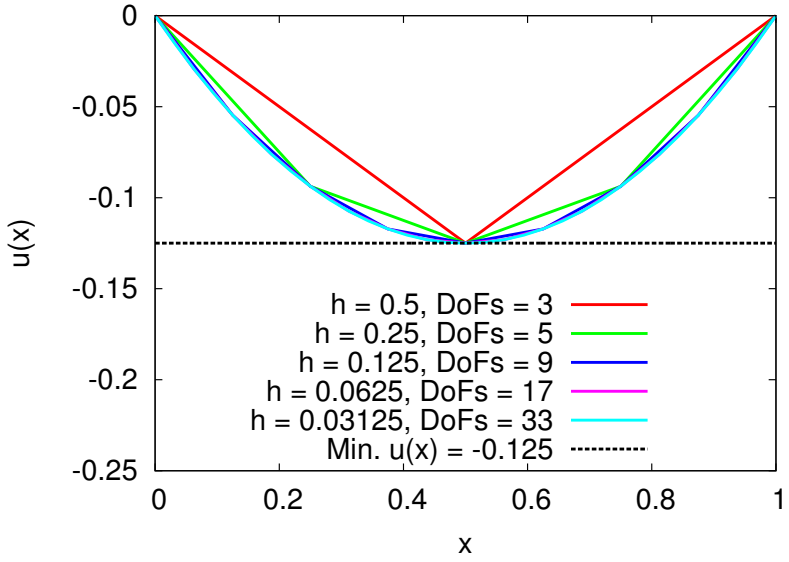


Figure 18: Solution of the 1D Poisson problem with $f = -1$ using finite elements with various mesh sizes h. DoFs is the abbreviation for degrees of freedom; here the number of support points x_j. The dimension of the discrete space is $DoFs$. For instance for $h = 0.5$, we have 3 DoFs and two basis functions, thus $dim(V_h) = 3$. The numerical solutions are computed with an adaptation of step-3 in deal.II [6, 9]. Please notice that the picture norm is not a proof in the strict mathematical sense: to show that the purple, and blue lines come closer and closer must be confirmed by error estimates as presented in Section 8.13 accompanied by numerical simulations in Section 8.16. Of course, for this 1D Poisson problem, we easily observe a limit case, but for more complicated equations it is often not visible whether the solutions do converge.

```
Level  Elements   DoFs
======================
1        2          3
2        4          5
3        8          9
4       16         17
5       32         33
======================
```

8.5 Algorithmic details

We provide some algorithmic details for implementing finite elements in a computer code. Of course, for constant coefficients, a 'nice' domain Ω with uniform step lengths, we can immediately build the matrix, right hand side vector and compute the solution. In practice, we are interested in more general formulations. Again, we explain all details in 1D, and partially follow Suttmeier's lecture [124]. Further remarks and details (in particular for higher dimensions) can be found in the literature provided in Section 1.

8.5.1 Assembling integrals on each cell K

In practice, one does not proceed as in Section 8.4.5 since this only makes sense for very simple problems.

A more general procedure is based on an element-wise consideration, which is motivated by:

$$a(u,\phi) = \int_\Omega u'\,\phi'\,dx = \sum_{K\in\mathcal{T}_h} \int_K u'\,\phi'\,dx.$$

The basic procedure is:

Algorithm 8.36 (Basic assembling - robust, but partly inefficient)**.** *Let $K_s, s = 0,\dots n$ be an element and let i and j be the indices of the degrees of freedom (namely the basis functions). The basic algorithm to compute all entries of the system matrix and right hand side vector is:*

$$\begin{aligned}
&\textit{for all elements } K_s \textit{ with } s = 0,\dots,n\\
&\quad \textit{for all DoFs } i \textit{ with } i = 0,\dots,n+1\\
&\quad\quad \textit{for all DoFs } j \textit{ with } j = 0,\dots,n+1\\
&\quad\quad\quad a_{ij} += \int_{K_s} \phi_i'(x)\,\phi_j'(x)\,dx
\end{aligned}$$

Here $+=$ means that entries with the same indices i, j are summed. For the right hand side, we have

$$\begin{aligned}
&\textit{for all elements } K_s \textit{ with } s = 0,\dots,n\\
&\quad \textit{for all DoFs } i \textit{ with } i = 0,\dots,n+1\\
&\quad\quad b_i += \int_{K_s} f(x)\phi_i(x)\,dx.
\end{aligned}$$

Again $+=$ means that only the b_i with the same i are summed.

Remark 8.37. *This algorithm is a bit inefficient since a lot of zeros are added. Knowing in advance the polynomial degree of the shape functions allows to add an if-condition to assemble only non-zero entries.*

8.5.2 Example in $\mathbb{R}^5$

We illustrate the previous algorithm for a concrete example. Let us compute 1D Poisson on four support points $x_i, i = 0,1,2,3,4$ that are equidistantly distributed yielding a uniform mesh size $h = x_j - x_{j-1}$. The discrete space V_h is given by:

$$V_h = \{\phi_0,\phi_1,\phi_2,\phi_3,\phi_4\}, \quad dim(V_h) = 5.$$

The number of cells is $\#K = 4$.

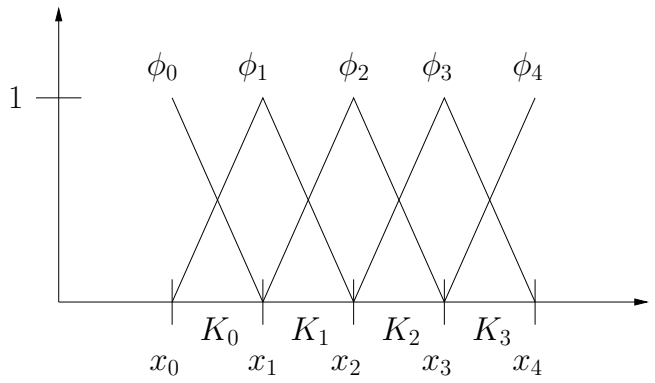


Figure 19: Basis functions ϕ_i, elements K_s and nodes x_s used in Section 8.5.2.

It holds furthermore:

$$U \in \mathbb{R}^5, \quad A \in \mathbb{R}^{5\times 5}, \quad b \in \mathbb{R}^5.$$

8.5.2.1 Element K_0 We start with $s = 0$, namely K_0:

$$a_{00}^{s=0} = a_{00} = \int_{K_0} \phi_0'\phi_0' = \frac{1}{h}, \quad a_{01}^{s=0} = a_{01} = \int_{K_0} \phi_0'\phi_1' = -\frac{1}{h},$$
$$a_{02}^{s=0} = a_{02} = \int_{K_0} \phi_0'\phi_2' = 0, \quad a_{03}^{s=0} = a_{03} = \int_{K_0} \phi_0'\phi_3' = 0, \quad a_{04}^{s=0} = a_{04} = \int_{K_0} \phi_0'\phi_4' = 0.$$

Similarly, we evaluate $a_{1j}, a_{2j}, a_{3j}, a_{4j}, j = 0, \ldots 4$. This yields the local matrix:

$$\tilde{A}^{s=0} = \frac{1}{h}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

and the global matrix

$$A^{s=0} = \frac{1}{h}\begin{pmatrix} +1 & -1 & 0 & 0 & 0 \\ -1 & +1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

We also see that we add a lot of zeros when $|i - j| > 1$. For this reason, a good algorithm first design the sparsity pattern and determines the entries of A that are non-zero. This is clear due to the construction of the hat functions and that they only overlap on neighboring elements.

8.5.2.2 Element K_1 Next, we increment $s = 1$ and work on cell K_1. Here we again evaluate all a_{ij} and sum them to the previously obtained values on K_0. Therefore the $+=$ in the above algorithm.

This yields the local matrix:

$$\tilde{A}^{s=1} = \frac{1}{h}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

and the global matrix

$$A^{s=1} = \frac{1}{h}\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & +1 & -1 & 0 & 0 \\ 0 & -1 & +1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

8.5.2.3 Assembling all elements After having assembled the values on all four elements $K_s, s = 1, 2, 3, 4$, we obtain the following system matrix:

$$A = \begin{pmatrix} \sum_s a^s_{00} & \sum_s a^s_{01} & \sum_s a^s_{02} & \sum_s a^s_{03} & \sum_s a^s_{04} \\ \sum_s a^s_{10} & \sum_s a^s_{11} & \sum_s a^s_{12} & \sum_s a^s_{13} & \sum_s a^s_{14} \\ \sum_s a^s_{20} & \sum_s a^s_{21} & \sum_s a^s_{22} & \sum_s a^s_{23} & \sum_s a^s_{24} \\ \sum_s a^s_{30} & \sum_s a^s_{31} & \sum_s a^s_{32} & \sum_s a^s_{33} & \sum_s a^s_{34} \\ \sum_s a^s_{40} & \sum_s a^s_{41} & \sum_s a^s_{42} & \sum_s a^s_{43} & \sum_s a^s_{44} \end{pmatrix} = \sum_{s=0}^{3} A^{(s)} = \frac{1}{h} \begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 1+1 & -1 & 0 & 0 \\ 0 & -1 & 1+1 & -1 & 0 \\ 0 & 0 & -1 & 1+1 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

To fix the homogeneous Dirichlet conditions, we can manipulate directly the matrix A or work with a 'constraint matrix'. We eliminate the entries of the rows and columns of the off-diagonals corresponding to the boundary indices; here $i = 0$ and $i = 4$. Then:

$$A = \frac{1}{h} \begin{pmatrix} h & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & h \end{pmatrix}.$$

8.5.3 On the indices of A

We investigate whether

$$a_{ij} = (\nabla\phi_j, \nabla\phi_i) = (\nabla\phi_i, \nabla\phi_j).$$

For $(\nabla u, \nabla\phi)$ this result is true because of symmetry. For non-symmetric problems this result is not anymore true. Consider for instance

$$(\nabla u, \nabla\phi) + (b\nabla u, \phi),$$

where $b \in \mathbb{R}^d$. Take now

$$u_h = \sum_{j=1}^{n} u_j \phi_j$$

and insert for each test function $\phi_i, i = 1, \ldots, n$:

$$\begin{aligned} &\sum_{j=1}^{n} u_j[(\nabla\phi_j, \nabla\phi_1) + (b\nabla\phi_j, \phi_1)], \quad i = 1, \\ &\vdots \\ &\sum_{j=1}^{n} u_j[(\nabla\phi_j, \nabla\phi_n) + (b\nabla\phi_j, \phi_n)], \quad i = n, \end{aligned}$$

As we already know and again see is that the test function determines the row in the system matrix A. For this reason, on the right hand side, the first index j determines the column and the second index i determines the row:

$$a_{ij} = \sum_{j=1}^{n} u_j[(\nabla\phi_j, \nabla\phi_i) + (b\nabla\phi_j, \phi_i)]$$

Since this problem is non-symmtric, we have $a_{ij} \neq a_{ji}$ because $(b\nabla\phi_j, \phi_i) \neq (b\nabla\phi_i, \phi_j)$. Here is a typical code snippet from deal.II:

```
for (unsigned int i=0; i<dofs_per_cell; ++i)
{
 ...
  for (unsigned int j=0; j<dofs_per_cell; ++j)
```

```
  {
    local_matrix(j,i) += (b * phi_grads_u[i] * phi_u[j] +
                          scalar_product(phi_grads_u[i], phi_grads_u[j])
                         ) * fe_values.JxW(q);
  }
}
```

8.5.4 Numerical quadrature

As previously stated, the arising integrals may easily become difficult such that a direct integration is not possible anymore:

- Non-constant right hand sides $f(x)$ and non-constant coefficients $\alpha(x)$;
- Higher-order shape functions;
- Non-uniform step sizes, more general domains.

In modern FEM programs, Algorithm 8.36 is complemented by an alterative evaluation of the integrals using numerical quadrature. The general formula reads:

$$\int_\Omega g(x) \approx \sum_{l=0}^{n_q} \omega_l g(q_l)$$

with quadrature weights ω_l and quadrature points q_l. The number of quadrature points is $n_q + 1$.

Remark 8.38. *The support points x_i of the FE mesh and the quadrature points q_l do not need to be necessarily the same. For Gauss quadrature, they are indeed different. For lowest-order interpolatory quadrature rules (box, Trapez) they correspond though.*

We continue the above example by choosing the trapezoidal rule, which in addition, should integrate the arising integrals exactly:

$$\int_{K_s} g(x) \approx h_s \sum_{l=0}^{n_q} \omega_l g(q_l)$$

where h_s is the length of interval/element K_s, $n_q = 1$ and $\omega_l = 0.5$. This brings us to:

$$\int_{K_s} g(x) \approx h_s \frac{g(q_0) + g(q_1)}{2}.$$

Applied to our matrix entries, we have on an element K_s:

$$a_{ii} = \int_{K_s} \phi_i'(x)\phi_i'(x)\,dx \approx \frac{h_s}{2}\Big(\phi_i'(q_0)\phi_i'(q_0) + \phi_i'(q_1)\phi_i'(q_1)\Big).$$

For the right hand side, we for the case $f = 1$ we can use for instance the mid-point rule:

$$\int_{K_i} \phi_i(x)\,dx \approx h_i \phi_i\left(\frac{x_i + x_{i-1}}{2}\right) = h_i \phi_i\left(\frac{x_i + x_{i-1}}{2}\right).$$

Remark 8.39. *If $f = f(x)$ with a dependency on x, we should use a quadrature formula that integrates the function $f(x)\phi_i(x)$ as accurate as possible. The mid-point rule would read:*

$$\int_{K_i} f(x)\phi_i(x)\,dx \approx h_i \left[f\left(\frac{x_i + x_{i-1}}{2}\right)\phi_i\left(\frac{x_i + x_{i-1}}{2}\right)\right].$$

Remark 8.40. *It is important to notice that the order of the quadrature formula must be sufficiently high since otherwise the quadrature error dominates the convergence behavior of the FEM scheme.*

We have now all ingredients to extend Algorithm 8.36:

Algorithm 8.41 (Assembling using the trapezoidal rule)**.** *Let $K_s, s = 0, \dots n$ be an element and let i and j be the indices of the degrees of freedom (namely the basis functions). The basic algorithm to compute all entries of the system matrix A and right hand side vector b is:*

$$\begin{aligned}
&\textit{for all elements } K_s \textit{ with } s = 0, \dots, n \\
&\quad \textit{for all DoFs } i \textit{ with } i = 0, \dots, n+1 \\
&\quad\quad \textit{for all DoFs } j \textit{ with } j = 0, \dots, n+1 \\
&\quad\quad\quad \textit{for all quad points } l \textit{ with } l = 0, \dots, n_q \\
&\quad\quad\quad\quad a_{ij} += \frac{h_s}{2} \phi_i'(q_l) \phi_j'(q_l)
\end{aligned}$$

where $n_q = 1$. Here $+=$ means that entries with the same indices are summed. This is necessary because on all cells K_s we assemble again a_{ij}.

$$\begin{aligned}
&\textit{for all elements } K_s \textit{ with } s = 0, \dots, n \\
&\quad \textit{for all DoFs } i \textit{ with } i = 0, \dots, n+1 \\
&\quad\quad \textit{for all quad points } l \textit{ with } l = 0, \dots, n_q \\
&\quad\quad\quad b_i += h_s f(q_l) \phi_i(q_l)
\end{aligned}$$

Remark 8.42. *In practice it is relatively easy to reduce the computational cost when a lot of zeros are computed. We know due to the choice of the finite element, which DoFs are active on a specific element and can assemble only these shape functions. For linear elements, we could easily add an if condition that only these a_{ij} are assembled when $|i - j| \leq 1$. We finally remark that these loops will definitely work, but there is a second aspect that needs to be considered in practice which is explained in Section 8.5.5.*

8.5.5 Details on the evaluation on the master element

In practice, all integrals are transformed onto a **master element** (or so-called **reference element**) and evaluated there. This has the advantage that

- we only need to evaluate once all basis functions;
- numerical integration formulae are only required on the master element;
- independence of the coordinate system. For instance quadrilateral elements in 2D change their form when the coordinate system is rotated [106].

The price to pay is to compute at each step a deformation gradient and a determinant, which is however easier than evaluating all the integrals.

We consider the (physical) element $K_i^{(h_i)} = [x_i, x_{i+1}], i = 0, \dots n$ and the variable $x \in K_i^{(h_i)}$ with and $h_i = x_{i+1} - x_i$. Without loss of generality, we work in the following with the first element $K_0^{(h_0)} = [x_0, x_1]$ and $h = h_0 = x_1 - x_0$ as shown in Figure 20. The generalization to s elements is briefly discussed in Section 8.5.6.

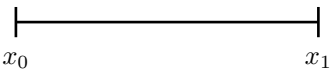


Figure 20: The mesh element $K_0^{(h_0)}$.

The element $K_i^{(h_i)}$ is transformed to the master element (i.e., the unit interval with mesh size $h = 1$) $K^{(1)} := [0, 1]$ with the local variable $\xi \in [0, 1]$ as shown in Figure 21.

For the transformations, we work with the substitution rule (see Section 3.15). Here in 1D, and in higher dimensions with the analogon. We define the mapping

$$\begin{aligned}
T_h : K^{(1)} &\to K_0^{(h_0)} \\
\xi &\mapsto T_h(\xi) = x = x_0 + \xi \cdot (x_1 - x_0) = x_0 + \xi h.
\end{aligned}$$

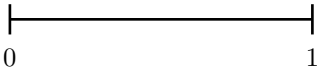


Figure 21: The master element K_1.

While function values can be identified in both coordinate systems, i.e.,

$$f(x) = \hat{f}(\xi), \quad \hat{f} \text{ defined in } K^{(1)},$$

derivatives will be complemented by further terms due to the chain rule that we need to employ. Differentiation in the physical coordinates yields

$$\begin{aligned} \frac{d}{dx}: \quad & 1 = (x_1 - x_0)\,\frac{d\xi}{dx} \\ \Rightarrow \quad & dx = (x_1 - x_0)\cdot d\xi. \end{aligned}$$

The volume (here in 1D: length) change can be represented by the determinant of the Jacobian of the transformation:

$$J := x_1 - x_0 = h.$$

These transformations follow exactly the way as they are known in continuum mechanics. Thus for higher dimensions we refer exemplarily to [76], where transformations between different domains can be constructed.

We now construct the inverse mapping

$$\begin{aligned} T_h^{-1} : K_0^{(h_0)} &\to K^{(1)} \\ x &\mapsto T_h^{-1}(x) = \xi = \frac{x - x_0}{x_1 - x_0} = \frac{x - x_0}{h}, \end{aligned}$$

with the derivative

$$\partial_x T_h^{-1}(x) = \xi_x = \frac{d\xi}{dx} = \frac{1}{x_1 - x_0}.$$

A basis function φ_i^h on $K_0^{(h_0)}$ reads:

$$\varphi_i^h(x) := \varphi_i^1(T_h^{-1}(x)) = \varphi_i^1(\xi)$$

and for the derivative we obtain with the chain rule:

$$\partial_x \varphi_i^h(x) = \partial_\xi \varphi_i^1(\xi) \cdot \partial_x T_h^{-1}(x) = \partial_\xi \varphi_i^1(\xi) \cdot \xi_x$$

with $T_h^{-1}(x) = \xi$.

Example 8.43. *We provide two examples. Firstly:*

$$\int_{K_h} f(x)\,\varphi_i^h(x)\,dx \overset{Sub.}{=} \int_{K^{(1)}} f(T_h(\xi)) \cdot \varphi_i^1(\xi) \cdot J \cdot d\xi, \tag{102}$$

and secondly,

$$\int_{K_h} \partial_x \varphi_i^h(x) \cdot \partial_x \varphi_j^h(x)\,dx = \int_{K^{(1)}} \Big(\partial_\xi \varphi_i^1(\xi)\Big) \cdot \xi_x \cdot \Big(\partial_\xi \varphi_j^1(\xi)\Big) \cdot \xi_x \cdot J\,d\xi. \tag{103}$$

We can now apply numerical integration using again the trapezoidal rule and obtain for the two previous integrals:

$$\int_{T_h} f(x)\,\varphi_i^h(x)\,dx \overset{(102)}{\approx} \sum_{k=1}^{q} \omega_k f(F_h(\xi_k))\,\varphi_i^1(\xi_k) \cdot J$$

and for the second example:

$$\int_{T_h} \partial_x \varphi_j^h(x) \partial_x \varphi_i^h(x)\,dx \approx \sum_{k=1}^{q} \omega_k \Big(\partial_\xi \varphi_j^1(\xi_k) \cdot \xi_x\Big) \cdot \Big(\partial_\xi \varphi_i^1(\xi_k) \cdot \xi_x\Big) \cdot J.$$

Remark 8.44. *These final evaluations can again be realized by using Algorithm 8.41, but are now performed on the unit cell $K^{(1)}$.*

8.5.6 Generalization to s elements

We briefly setup the notation to evaluate the integrals for s elements:

- Let n be the index of the end point $x_n = b$ (b is the nodal point of the right boundary). Then, $n-1$ is the number of elements (intervals in 1d), and $n+1$ is the number of the nodal points (degrees of freedom - DoFs) and the number shape functions, respectively (see also Figure 19);
- $K_s^{(h_s)} = [x_s, x_{s+1}], s = 0, \dots n-1.$
- $h_s = x_{s+1} - x_s$;
- $T_s : K^{(1)} \to K_s^{(h_s)} : \xi \mapsto T_s(\xi) = x_s + \xi(x_{s+1} - x_s) = x_s + \xi h_s$;
- $T_s^{-1} : K_s^{(h_s)} \to K^{(1)} : x \mapsto T_s^{-1}(x) = \frac{x - x_s}{h_s}$;
- $\nabla T_s^{-1}(x) = \partial_x T_s^{-1}(x) = \frac{1}{h_s}$ (in 1D);
- $\nabla T_s(\xi) = \partial_x T_s(\xi) = (x_s + \xi h_s)' = h_s$ (in 1D);
- $J_s := \det(\nabla T_s(\xi)) = (x_s + \xi h_s)' = h_s$ (in 1D).

8.5.7 Example: Section 8.5.2 continued using Sections 8.5.5 and 8.5.6

We continue Example 8.5.2 and build the system matrix A by using the master element. In the following, we evaluate the Laplacians in weak form:

$$\begin{aligned} a_{ij}^s = \int_{K_s} \phi_i'(x) \cdot \phi_j'(x)\, dx &= \int_{K^{(1)}} \phi_i'(\xi)\, \partial_x T_s^{-1}(x) \cdot \phi_j'(\xi)\, \partial_x T_s^{-1}(x)\, J_s\, d\xi \\ &= \int_{K^{(1)}} \phi_i'(\xi)\, \frac{1}{h_s} \cdot \phi_j'(\xi)\, \frac{1}{h_s}\, h_s\, d\xi \\ &= \int_{K^{(1)}} \phi_i'(\xi) \cdot \phi_j'(\xi)\, \frac{1}{h_s}\, d\xi. \end{aligned}$$

We now compute **once!** the required integrals on the unit element (i.e., the master element) $K^{(1)}$. Here, $\xi_0 = 0$ and $\xi_1 = 1$ with $h^{(1)} = \xi_1 - \xi_0 = 1$ resulting in two shape functions:

$$\phi_0(\xi) = \frac{\xi_1 - \xi}{h^{(1)}}, \qquad \phi_1(\xi) = \frac{\xi - \xi_0}{h^{(1)}}.$$

The derivatives are given by:

$$\phi_0'(\xi) = \frac{-1}{h^{(1)}} = -1, \qquad \phi_1'(\xi) = \frac{1}{h^{(1)}} = 1.$$

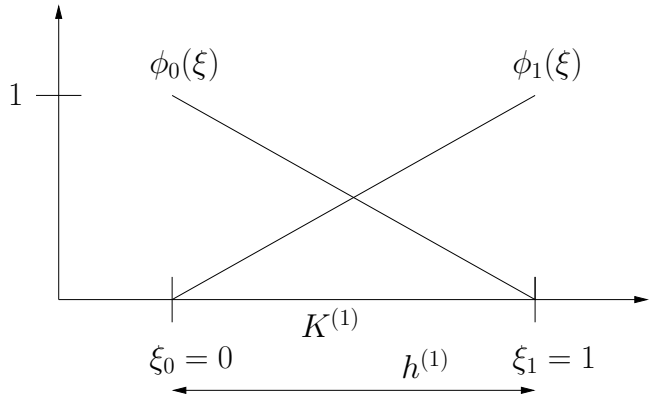


Figure 22: The master element $K^{(1)}$ including nodal points and mesh size parameter $h^{(1)}$.

With these calculations, we compute all combinations on the master element required to evaluate the Laplacian in 1d:

$$\begin{aligned}
a_{00}^{(1)} &= \int_{K^{(1)}} \phi_0'(\xi) \cdot \phi_0'(\xi)\, d\xi = \int_{K^{(1)}} (-1)^2\, d\xi = 1,\\
a_{01}^{(1)} &= \int_{K^{(1)}} \phi_0'(\xi) \cdot \phi_1'(\xi)\, d\xi = \int_{K^{(1)}} (-1)\cdot 1\, d\xi = -1,\\
a_{10}^{(1)} &= \int_{K^{(1)}} \phi_1'(\xi) \cdot \phi_0'(\xi)\, d\xi = \int_{K^{(1)}} 1\cdot (-1)\, d\xi = -1,\\
a_{11}^{(1)} &= \int_{K^{(1)}} \phi_1'(\xi) \cdot \phi_1'(\xi)\, d\xi = \int_{K^{(1)}} 1\cdot 1\, d\xi = 1.
\end{aligned}$$

It is clear what happens on each element K_s using Algorithm 8.36:

$$\begin{aligned}
a_{00}^{s} &= \int_{K^{(1)}} \phi_0'(\xi) \cdot \phi_0'(\xi)\, \frac{1}{h_s}\, d\xi = \frac{1}{h_s}\int_{K^{(1)}} (-1)^2\, d\xi = \frac{1}{h_s},\\
a_{01}^{s} &= \int_{K^{(1)}} \phi_0'(\xi) \cdot \phi_1'(\xi)\, \frac{1}{h_s}\, d\xi = \frac{1}{h_s}\int_{K^{(1)}} (-1)\cdot 1\, d\xi = \frac{-1}{h_s},\\
a_{02}^{s} &= \int_{K^{(1)}} \phi_0'(\xi) \cdot \phi_2'(\xi)\, \frac{1}{h_s}\, d\xi = \frac{1}{h_s}\int_{K^{(1)}} (-1)\cdot 0\, d\xi = 0,\\
a_{03}^{s} &= \int_{K^{(1)}} \phi_0'(\xi) \cdot \phi_3'(\xi)\, \frac{1}{h_s}\, d\xi = \frac{1}{h_s}\int_{K^{(1)}} (-1)\cdot 0\, d\xi = 0,\\
a_{04}^{s} &= \int_{K^{(1)}} \phi_0'(\xi) \cdot \phi_4'(\xi)\, \frac{1}{h_s}\, d\xi = \frac{1}{h_s}\int_{K^{(1)}} (-1)\cdot 0\, d\xi = 0.
\end{aligned}$$

Next, we increase $i \to i+1$ resulting in $i = 1$ and compute:

$$a_{10}^{s}, \quad a_{11}^{s}, \quad a_{12}^{s}, \quad a_{13}^{s}, \quad a_{14}^{s}.$$

We proceed until $i = 4$. This procedure is done for all elements K_s with $s = 0, 1, 2, 3$. As stated in Algorithm 8.36, the procedure is however inefficient since a lot of zeros are assembled. Knowing the structure of the matrix (i.e., the sparsity pattern) allows us upfront only to assemble the entries with non-zero entries.

Anyhow, the system matrix is composed by (the same as in Section 8.5.2):

$$\begin{aligned}
A &= \begin{pmatrix}
\sum_s a_{00}^s & \sum_s a_{01}^s & \sum_s a_{02}^s & \sum_s a_{03}^s & \sum_s a_{04}^s \\
\sum_s a_{10}^s & \sum_s a_{11}^s & \sum_s a_{12}^s & \sum_s a_{13}^s & \sum_s a_{14}^s \\
\sum_s a_{20}^s & \sum_s a_{21}^s & \sum_s a_{22}^s & \sum_s a_{23}^s & \sum_s a_{24}^s \\
\sum_s a_{30}^s & \sum_s a_{31}^s & \sum_s a_{32}^s & \sum_s a_{33}^s & \sum_s a_{34}^s \\
\sum_s a_{40}^s & \sum_s a_{41}^s & \sum_s a_{42}^s & \sum_s a_{43}^s & \sum_s a_{44}^s
\end{pmatrix}\\
&= \begin{pmatrix}
\frac{1}{h_0} & \frac{-1}{h_0} & 0 & 0 & 0 \\
\frac{-1}{h_0} & \frac{1}{h_0}+\frac{1}{h_1} & \frac{-1}{h_1} & 0 & 0 \\
0 & \frac{-1}{h_1} & \frac{1}{h_1}+\frac{1}{h_2} & \frac{-1}{h_2} & 0 \\
0 & 0 & \frac{-1}{h_2} & \frac{1}{h_2}+\frac{1}{h_3} & \frac{-1}{h_3} \\
0 & 0 & 0 & \frac{-1}{h_3} & \frac{1}{h_3}
\end{pmatrix}.
\end{aligned}$$

This matrix is the same as if we had evaluated all shape functions on the physical elements with a possibly non-uniform step size $h_s, s = 0, 1, 2, 3$.

Remark 8.45. *Good practical descriptions for higher dimensions can be found in [82][Chapter 12] and [120].*

8.6 Quadratic finite elements: P_2 elements

8.6.1 Algorithmic aspects

We explain the idea how to construct higher-order FEM in this section. Specifically, we concentrate on P_2 elements, which are quadratic on each element K_s. First we define the discrete space:

$$V_h = \{v \in C[0,1] |\ v|_{K_j} \in P_2\}$$

The space V_h is composed by the basis functions:

$$V_h = \{\phi_0, \ldots, \phi_{n+1}, \phi_{\frac{1}{2}}, \ldots, \phi_{n+\frac{1}{2}}\}.$$

The dimension of this space is $\dim(V_h) = 2n+1$. Here, we followed the strategy that we use the elements K_s as in the linear case and add the mid-points in order to construct unique parabolic functions on each K_s. The mid-points represent degrees of freedom as the two edge points. For instance on each $K_j = [x_j, x_{j+1}]$ we have as well $x_{j+\frac{1}{2}} = x_j + \frac{h}{2}$, where $h = x_{j+1} - x_j$. From this construction it is clear (not proven though!) that quadratic FEM have a higher accuracy than linear FEM.

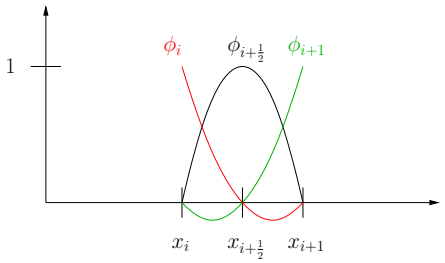


Figure 23: Quadratic basis functions

The specific construction of shape functions can be done as shown in 8.4.3 or using directly Lagrange basis polynomials (see lectures on introduction of numerical methods).

Definition 8.46 (P_2 shape functions)**.** *On the master element $K^{(1)}$, we have*

$$\phi_0(\xi) = 1 - 3\xi + 2\xi^2,$$
$$\phi_{\frac{1}{2}}(\xi) = 4\xi - 4\xi^2,$$
$$\phi_1(\xi) = -\xi + 2\xi^2.$$

These basis functions fulfill the property:

$$\phi_i(\xi_j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

for $i, j = 0, \frac{1}{2}, 1$. On the master element, a function has therefore the prepresentation:

$$u(\xi) = \sum_{j=0}^{1} u_j \phi_j(\xi) + u_{\frac{1}{2}} \phi_{\frac{1}{2}}(\xi).$$

This definition allows us to construct a global function from V_h:

Proposition 8.47. *The space V_h is a subspace of V. Each function from V_h has a unique representation and is defined by its nodal points:*

$$u_h(x) = \sum_{j=0}^{n+1} u_j \phi_j(x) + \sum_{j=0}^{n} u_{j+\frac{1}{2}} \phi_{\frac{1}{2}}(x).$$

Remark 8.48. *The assembling of A, u and b is done in a similar fashion as shown in detail for linear finite elements.*

8.6.2 Numerical test: 1D Poisson using quadratic FEM (P_2 elements)

We continue Section 8.4.9.

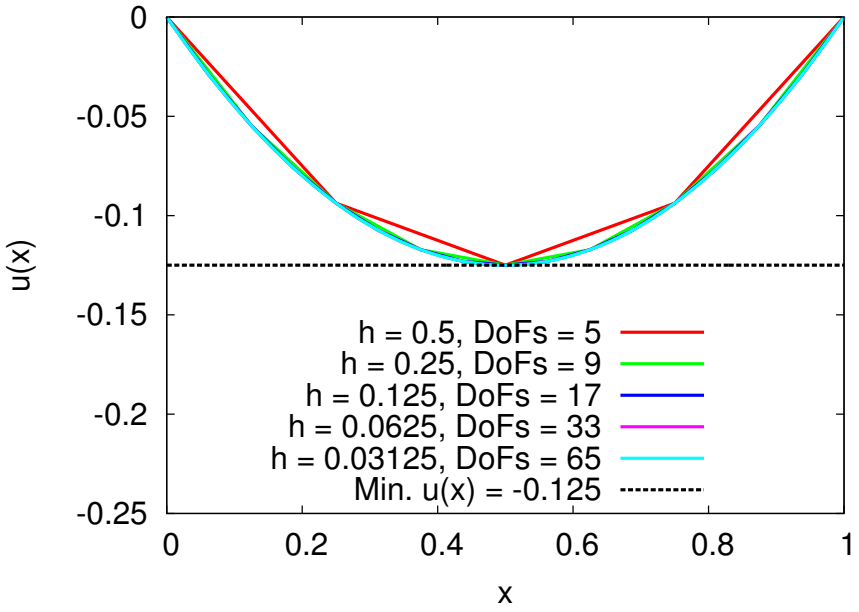


Figure 24: Solution of the 1D Poisson problem with $f = -1$ using quadratic finite elements with various mesh sizes h. DoFs is the abbreviation for degrees of freedom; here the number of support points x_j. The dimension of the discrete space is $DoFs$. For instance for $h = 0.5$, we have 2 mesh elements, we have 5 DoFs and three basis functions, thus $dim(V_h) = 5$. The numerical solutions are computed with an adaptation of step-3 in deal.II [6, 9]. Please notice that the picture norm is not a proof in the strict mathematical sense: to show that the purple, and blue lines come closer and closer must be confirmed by error estimates as presented in Section 8.13 accompanied by numerical simulations in Section 8.16. Of course, for this 1D Poisson problem, we easily observe a limit case, but for more complicated equations it is often not visible whether the solutions do converge.

Remark 8.49. *Be careful when plotting the solution using higher-order FEM. Sometimes, the output data is only written into the nodal values values defining the physical elements. In this case, a quadratic function is visually the same as a linear function. Plotting in all degrees of freedom would on the other hand shows also visually that higher-order FEM are employed.*

Level	Elements	DoFs
1	2	5
2	4	9
3	8	17
4	16	33
5	32	65

8.7 Galerkin orthogonality, a geometric interpretation of the FEM, and a first error estimate

The finite element method has a remarkable geometric interpretation, which is finally the key how to solve and analyze a given PDE in an infinite-dimensional vector space V using the finite-dimensional space V_h as we have previously constructured from a practical point of view. The techniques are based on results from functional analysis.

The goal is to study the accuracy of our FEM scheme in terms of the discretization error $u - u_h$. Here, $u \in V$ is the exact (unknown) solution and $u_h \in V_h$ our finite element solution. The key is to use first **best approximation** results and later (see the later sections) **interpolation estimates**.

We have

$$(u', \phi') = (f, \phi) \quad \forall \phi \in V,$$
$$(u_h', \phi_h') = (f, \phi_h) \quad \forall \phi_h \in V_h.$$

Taking in particular only discrete test functions from $V_h \subset V$ and subtraction of both equations yields:

Proposition 8.50 (Galerkin orthogonality)**.** *It holds:*

$$((u - u_h)', \phi_h') = 0 \quad \forall \phi_h \in V_h,$$

or in the more general notation:

$$a(u - u_h, \phi_h) = 0 \quad \forall \phi_h \in V_h.$$

This also shows that we deal with a projection.

Proof. Taking $\phi_h \in V_h$ in both previous equations yields:

$$(u', \phi_h') - (u_h', \phi_h') = (f, \phi_h) - (f, \phi_h).$$

Taking both equations, in the discrete space V_h yields

$$(f, \phi_h) - (f, \phi_h) = 0.$$

□

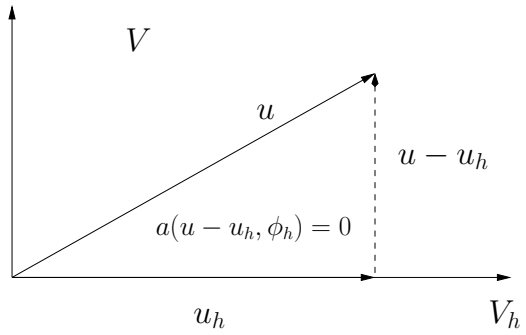


Figure 25: Illustration of the Galerkin orthogonality.

Since $(\cdot, \cdot)$ is a scalar product (for the definition we refer the reader to Definition 7.7); here in the L^2 sense, [6] Galerkin orthogonality yields immediately a geometric interpretation of the finite element method. The error (measured in terms of the first derivative) stands orthogonal to all elements ϕ_h of V_h. Therefore, the solution u_h is the **best approximation** since it has the smallest distance to u.

[6] We later define more precisely the L^2 space. The reader can think of a square-integrable functions defined on Ω with

$$L^2(\Omega) := \{v | \int_\Omega v^2 \, dx < \infty\}.$$

Proposition 8.51 (Best approximation in finite-dimensional subspaces)**.** *Let U be a finite-dimensional subspace of V (for instance $U := V_h$, our finite element function space). Then, there exists for each element in V a best approximation in U.*

Proof. Let $u \in V$. Choosing a minimizing sequence[7] $(v_n)_{n\in\mathbb{N}}$ from U to approximate u brings us to

$$\|u - v_n\| \to d \quad \text{for } n \to \infty,$$

where $d := \inf_{v\in U} \|u - v\|$. Moreover, we can estimate as follows:

$$\begin{aligned}\|v_n\| &= \|v_n\| - \|u\| + \|u\| \\ &\leq |\|v_n\| - \|u\|| + \|u\| \\ &= |\|u\| - \|v_n\|| + \|u\| \\ &\leq \|u - v_n\| + \|u\|.\end{aligned}$$

Thus, the sequence $(v_n)_{n\in N}$ is bounded. Since U is finite dimensional, there exists a converging subsequence $(v_{n_l})_{l\in N}$ (a result from functional analysis, e.g., [134]) with

$$\lim_{l\to\infty} u_{n_l} \to v \in U.$$

It follows that

$$\|u - v\| = \lim_{l\to\infty} \|u - v_{n_l}\| = \inf_{\phi\in U} \|u - \phi\|$$

which shows the assertion. □

In the following, we use the fact that the L^2 scalar product induces a norm (this is due to the fact that L^2 is a Hilbert space; see Section 7.8)

$$\|w\| = \sqrt{(w, w)} = \sqrt{\int_\Omega w^2\, dx}.$$

We recall Cauchy's inequality, which we need in the following:

$$|(v, w)| \leq \|v\|\|w\|.$$

We show that u_h is a best approximation to u.

Theorem 8.52 (Céa lemma - first version)**.** *For any $v_h \in V_h$ it holds*

$$\|(u - u_h)'\| \leq \|(u - v_h)'\|$$

Proof. We estimate as follows:

$$\begin{aligned}\|(u - u_h)'\|^2 &= (u' - u_h', u' - u_h') \\ &= a(u - u_h, u - u_h) \\ &= a(u - u_h, u - v_h + v_h - u_h) \\ &= a(u - u_h, u - v_h) + \underbrace{a(u - u_h, v_h - u_h)}_{=0,\text{ Galerkin ortho.}} \\ &\leq \|(u - u_h)'\|\, \|(u - v_h)'\|.\end{aligned}$$

We can now divide by $|(u - u_h)'\|$ and obtain

$$\|(u - u_h)'\| \leq \|(u - v_h)'\|,$$

which shows the assertion. We finally remark that this proof technique (using Galerkin orthogonality at some point) is essential in finite element error analysis. □

[7] Working with minimizing sequences is an important concept in optimization and calculus of variations when minimizers to functionals need to be established. If (v_n) is a minimizing sequence and the functional $F(\cdot)$ is bounded from below, we obtain that $F(v_n) < \infty$ (assuming that F is proper) and $\lim_n F(v_n) = \inf_u F(u)$.

Remark 8.53. *This theorem is a first version of Céa's Lemma, which will be later augmented with certain constants yielding a more precise result. However, this result itself is extremely important as it shows the best approximation property: we choose the best approximation u_h from the space V_h. In particular, any interpolation error is based on the best approximation property. Specifically, v_h will be chosen as an appropriate interpolation $i_h v \in V_h$ from which we then can explicitely calculate the discrete functions and obtain a quantitative error estimate in terms of the mesh size h. Secondly, we also see (trivial, but nevertheless worth to be mentioned) that if u is already an element of V_h, the error would be identically zero, which is clear from geometrical considerations.*

The previous result can be extended as follows: not only does the Galerkin orthogonality yield the best approximation property, but it does also hold the contrary, namely that a best approximation property yields

$$((u - u_h)', \phi_h) = 0.$$

Proposition 8.54. *Let U be a linear subspace of a Pre-Hilbert space V. An element $v \in U$ is best approximation to $u \in V$ if and only if*

$$a(u - v, \phi) = 0 \quad \forall \phi \in U.$$

For each $u \in V$ there exists at most one best approximation with respect to U.

Proof. The backward part follows from Theorem 8.52. The forward direction follows from Section 7.3. □

8.8 Neumann & Robin problems

8.8.1 Robin boundary conditions

We briefly introduce and discuss well-posedness of Neumann and Robin boundary value problems. Let $\Omega = (0, 1)$ and let $f \in C(\Omega)$ and $\beta > 0$. We consider the following problem:

Formulation 8.55 ((D))**.** *Find $u \in C^2(\bar{\Omega})$ such that*

$$\begin{cases} -u'' = f & \text{in } \Omega, \\ \beta u(0) - u'(0) = \alpha_0, \\ \beta u(1) + u'(1) = \alpha_1. \end{cases} \tag{104}$$

Remark 8.56. *For $\beta = 0$ we would obtain a pure Neumann (or so-called traction) problem.*

To derive a variational formulation, a first step is the definition of an appropriate function space V. We recall that boundary conditions on function values (such as Dirichlet or the first terms $u(0)$ and $u(1)$ in the Robin type conditions) are built into V. If we change the problem statement (thus other boundary conditions), also the definition of V will change. Here, we define

$$V = \mathcal{C}^1(\Omega) \cap \mathcal{C}(\bar{\Omega}).$$

In the second step, we can state the variational form of (104):

Formulation 8.57 ((V))**.** *Find $u \in V$ such that*

$$a(u, \phi) = l(\phi) \quad \forall \phi \in V$$

where

$$\begin{aligned} a(u, \phi) &= \int_\Omega u'\phi' \, dx + \beta u(1)\phi(1) + \beta u(0)\phi(0), \\ l(\phi) &= \int_\Omega f\phi \, dx + \alpha_1 \phi(1) + \alpha_0 \phi(0). \end{aligned} \tag{105}$$

Proposition 8.58. *For $u \in \mathcal{C}^2(\bar{\Omega})$, it holds*

$$(D) \leftrightarrow (V).$$

Proof. First, we show $(D) \to (V)$. Multiplication with a test function, integration, and applying integration by parts yield

$$\begin{aligned} -u'' &= f \\ \Rightarrow (-u'', \phi) &= (f, \phi) \\ \Rightarrow (u', \phi') - [u'\phi]_0^1 &= (f, \phi) \\ \Rightarrow (u', \phi') - [u'(1)\phi(1) - u'(0)\phi(0)] &= (f, \phi) \\ \Rightarrow (u', \phi') - (\alpha_1 - \beta u(1))\phi(1) + (\beta u(0) - \alpha_0)\phi(0) &= (f, \phi) \\ \Rightarrow \underbrace{(u', \phi') + \beta u(1)\phi(1) + \beta u(0)\phi(0)}_{=a(u,\phi)} &= \underbrace{(f, \phi) + \alpha_1\phi(1) + \alpha_0\phi(0)}_{=l(\phi)}. \end{aligned}$$

We show the other direction $(V) \to (D)$. Let u be solution to the variational problem. After backward integration by parts, we obtain:

$$\begin{aligned} (-u'' - f, \phi) &= -[u'(x)\phi(x)]_{x=0}^{x=1} + [\alpha_0 - \beta u(0)]\phi(0) + [\alpha_1 - \beta u(1)]\phi(1) \\ &= [\alpha_0 + u'(0) - \beta u(0)]\phi(0) + [\alpha_1 - u'(1) - \beta u(1)]\phi(1) \end{aligned}$$

We work in a two-step procedure and discuss first the domain integral terms and in a second step the boundary terms. First, we work with the test space C_c^∞ (see Definition 8.10). Definitely $\phi \in C_c^\infty$, it follows $\phi \in C^1 \cap C^0$. We note that C_c^∞ is a dense subspace of V (see Definition 7.22) and therefore an admissible choice as test space. Using C_c^∞, the boundary terms vanish and we have

$$(-u'' - f, \phi) = 0 \quad \Rightarrow \quad -u'' - f = 0 \quad \Rightarrow \quad -u'' = f \quad \forall \phi \in C_c^\infty.$$

Since the domain terms fulfill the equation, the boundary terms remain:

$$[\alpha_0 + u'(0) - \beta u(0)]\phi(0) + [\alpha_1 - u'(1) - \beta u(1)]\phi(1) = 0.$$

We choose special test functions $\phi \in V$ with $\phi(0) = 1$ and $\phi(1) = 0$ yielding

$$\alpha_0 + u'(0) - \beta u(0) = 0$$

and thus the first boundary condition. Vice versa, we choose $\phi(1) = 0$ and $\phi(0) = 1$ and obtain

$$\alpha_1 - u'(1) - \beta u(1) = 0,$$

which is the second boundary condition. In summary, we have obtained the strong formulation and everything is shown. □

Proposition 8.59. *For $\beta > 0$ the solutions to (D) and (V) are unique.*

Proof. We assume that there are two solutions u_1 and u_2 and we define the difference function $w = u_1 - u_2$. Thanks to linearity it holds:

$$a(w, \phi) = 0 \quad \forall \phi \in V.$$

We recall that $a(\cdot, \cdot)$ has been defined in (105). As test function, we choose $\phi := w$:

$$a(w, w) = 0.$$

In detail:

$$\int_\Omega (w')^2 \, dx + \beta w(1)^2 + \beta w(0)^2 = 0.$$

Recalling that $\beta > 0$, the previous equation can only be satisfied when $w(1) = w(0) = \int_\Omega (w')^2 = 0$. In particular the integral yields that

$$w' \equiv 0 \quad \Rightarrow \quad w = \textit{constant}.$$

Since the solutions u_1 and u_2 are continuous (because we work with V - the reader may recall its definition!) the difference function w is continuous as well. Since on the boundary, we have $w(0) = w(1) = 0$, it follows from the continuity of w that necessarily

$$w \equiv 0 \quad \Rightarrow \quad u_1 - u_2 = 0.$$

□

8.8.2 Neumann boundary conditions

Working with Neumann boundary conditions, we have derivative information on the boundaries.

Formulation 8.60 ((D)). *Let $\Omega = (0,1)$. Find $u \in C^2(\bar{\Omega})$ such that*

$$\begin{cases} -u'' = f & \text{in } \Omega, \\ -u'(0) = \alpha_0, \\ u'(1) = \alpha_1. \end{cases} \tag{106}$$

The minus sign in front of $u'(0)$ is only for convenience.

Here, the solution is not unique anymore and only determined up to a constant:

$$u + c, \quad c > 0.$$

It is trivial to see that this solution satisfies Formulation 8.60. One possibility is to add an additional condition to fix the solution. A common choice is to prescribe the mean value:

$$\int_\Omega u(x)\,\mathrm{d}x = 0. \tag{107}$$

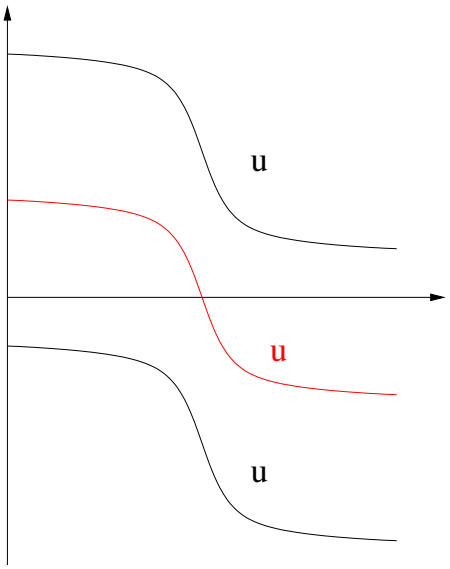


Figure 26: Sketch of possible solutions to the Poisson problem with Neumann boundary conditions. The red solution fulfills the compatibility condition (107). But any other constant to fix the solution would work as well such that one of the black solutions would be feasbile.

A variational formulation of (106) on $V = \mathcal{C}^1(\Omega) \cap \mathcal{C}(\bar{\Omega})$ reads:

Formulation 8.61 ((V)). *Find $u \in V$ such that*

$$a(u,\phi) = l(\phi) \quad \forall \phi \in V$$

where

$$\begin{aligned} a(u,\phi) &= \int_\Omega u'\phi'\,dx, \\ l(\phi) &= \int_\Omega f\phi\,dx + \alpha_1\phi(1) + \alpha_0\phi(0). \end{aligned} \tag{108}$$

Proposition 8.62. *If Formulation 8.60 has a solution $u \in \mathcal{C}^2(\bar{\Omega})$, it is unique by applying the normalization condition* (107)*. Moreover, a* ***compatibility condition****, which is sufficient and necessary, must hold:*

$$\int_0^1 f(x)\,\mathrm{d}x + \alpha_0 + \alpha_1 = 0.$$

Proof. More a sketch rather a rigorous result! It holds necessarily using integration by parts in 1D (Gauss divergence theorem to be more precise)

$$\int_\Omega f\,dx = -\int_\Omega u''\,dx = -\int_\Omega u'' \cdot 1\,dx = -[u'(x)]_0^1 = -[u'(1) - u'(0)] = -\alpha_0 - \alpha_1.$$

Consequently, we have shown the compatibility condition:

$$\int_\Omega f\,dx + \alpha_0 + \alpha_1 = 0.$$

This condition is also sufficient to obtain a solution:

$$\begin{aligned}
&\int_\Omega f\,dx + \alpha_0 + \alpha_1 = 0 \\
\Leftrightarrow &\int_\Omega f \cdot 1\,dx + \alpha_0 \cdot 1 + \alpha_1 \cdot 1 = 0 \\
\Leftrightarrow &\underbrace{\int_\Omega f\phi\,dx + \alpha_0 \cdot \phi + \alpha_1 \cdot \phi}_{=l(\phi)} = 0 \\
&\Leftrightarrow \underbrace{\int_\Omega u'\phi'}_{=a(u,\phi)} = 0 \quad \forall \phi = const.
\end{aligned}$$

Therefore, $a(u,\phi) = l(\phi)$. □

8.9 Variational formulations of elliptic problems in higher dimensions

In the previous sections of this chapter, we have restricted ourselves to 1D derivations in order to show the principle mechanisms. In the forthcoming sections, we extend everything to $\mathbb{R}^n$.

However, going from 1D to higher dimensions requires a **deeper mathematical theory** because the solutions to elliptic problems (e.g., Poisson) can be irregular such that strong solutions in classical function spaces may become infeasible. Consequently, we need to enlarge the sets of adequate solutions.

To this end, one restricts often to finding only weak solutions. For the integrals, however, the classical Riemann integral is again too restrictive. For this reason, the **Lebuesgue integral** must be adopted. We refer the reader to Section 7.1.3. A rich theory has been developed in the literature that is subject in classical lectures on the theory of PDEs and also functional analysis. We have gathered the most important aspects in Chapter 7.

In all this, principle challenges are:

1. Singularities in the domain (not a problem in 1D!), in the equations, the data become more severe in higher dimensions;

2. The natural function spaces (Sobolev spaces), here $H^1(\Omega)$ with $\Omega \subset \mathbb{R}^d, d \geq 2$, for solving the weak problem cannot be embedded into classical continuous spaces. This causes challenges in existence theories (for linear problems Lax-Milgram can be adopted) and also the prescription of boundary and initial data. These have to be formulated and interpreted in the Lebuesgue intergral sense.

8.9.1 Model problem

The model problem is

Formulation 8.63 (Model problem)**.** *Let $\Omega \subset \mathbb{R}^n$. Find $u \in C^2 \cap C(\bar{\Omega})$ such that*

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega \\ u &= 0 \quad \text{on } \partial\Omega. \end{aligned}$$

The Laplace-Operator Δ has been previously defined in Definition 3.4.

8.9.2 Comments on the domain Ω and its boundaries $\partial\Omega$

We recall the basic definitions of a domain and its boundaries. Let $\Omega \subset \mathbb{R}^n$ be a domain (its properties are specified in a minute) and $\partial\Omega$ its boundary. The closure of Ω is denoted by $\bar{\Omega}$. The boundary may be split into non-overlapping parts

$$\partial\Omega = \partial\Omega_D \cup \partial\Omega_N \cup \partial\Omega_R,$$

where $\partial\Omega_D$ represents a Dirichlet boundary, $\partial\Omega_N$ represents a Neumann boundary and $\partial\Omega_R$ represents a Robin boundary.

- By $C(\Omega)$ we denote the function space of continuous functions in Ω.
- Let $k \geq 0$ (an integer). Then we denote by $C^k(\Omega)$ the functions that are k-times continuously differentiable in Ω.
- By $C^\infty(\Omega)$ we denote the space of infinitely differentiable functions.
- By $C_c^\infty(\Omega)$ we denote the space of infinitely differentiable functions with compact support.

In order to be able to define a normal vector n, we assume that Ω is sufficiently regular, here Ω is of class C^1 (or we also say $\partial\Omega \in C^1$ or Ω has a Lipschitz boundary). The normal vector points outward of the domain.

Remark 8.64. *Depending on the properties of Ω (bounded or not), its regularity, and also the properties of its boundary, one can obtain very delicate mathematical results. We refer exemplarily to Grisvard [63] or Wloka [145]. We always assume in the remainder that Ω is open and bounded and sufficiently regular.*

8.9.3 Integration by parts and Green's formulae

From the divergence Theorem 3.33, we obtain immediately:

Proposition 8.65 (Integration by parts)**.** *Let $u, v \in C^1(\bar{\Omega})$. Then:*

$$\int_\Omega u_{x_i} v \, dx = -\int_\Omega u v_{x_i} \, dx + \int_{\partial\Omega} u v n_i \, ds, \quad \text{for } i = 1, \ldots, n.$$

In compact notation:

$$\int_\Omega \nabla u v \, dx = -\int_\Omega u \nabla v \, dx + \int_{\partial\Omega} u v n \, ds.$$

Proof. Apply the divergence theorem to uv. Exercise. □

We obtain now some further results, which are very useful, but all are based directly on the integration by parts. For this reason, it is more important to know the divergence theorem and integration by parts formula.

Proposition 8.66 (Green's formulas)**.** *Let $u, v \in C^2(\bar{\Omega})$. Then:*

$$\begin{aligned} \int_\Omega \Delta u \, dx &= \int_{\partial\Omega} \partial_n u \, ds, \\ \int_\Omega \nabla u \cdot \nabla v \, dx &= -\int_\Omega \Delta u \, v \, dx + \int_{\partial\Omega} v \, \partial_n u \, ds. \end{aligned}$$

Proof. Apply integration parts. □

8.9.4 Variational formulations

We recall the details of the notation that we shall use in the following. In $\mathbb{R}^n$, it holds:

$$(\nabla u, \nabla \phi) = \int_\Omega \nabla u \cdot \nabla \phi \, dx = \int_\Omega \begin{pmatrix} \partial_1 u \\ \vdots \\ \partial_n u \end{pmatrix} \cdot \begin{pmatrix} \partial_1 \phi \\ \vdots \\ \partial_n \phi \end{pmatrix} dx = \int_\Omega \Big(\partial_1 u \partial_1 \phi + \ldots + \partial_n u \partial_n \phi \Big) \, dx.$$

We have

Proposition 8.67. *Let $u \in C^2(\bar{\Omega})$ and $f \in C(\bar{\Omega})$. Let V defined by*

$$V = \{\phi \in C^1(\bar{\Omega}) |\ \phi = 0 \text{ on } \partial\Omega\}.$$

The function u is a solution to (D) if and only if $u \in V$ such that

$$(\nabla u, \nabla \phi) = (f, \phi) \quad \forall \phi \in V.$$

Proof. The proof is the same as in the 1D case. Let u be the solution of (D). We multiply the PDE by a test function $\phi \in V$, integrate and perform integration by parts:

$$-(\Delta u, \phi) = -\int_{\partial\Omega} \partial_n u \phi \, ds + (\nabla u, \nabla \phi)$$

where $\partial_n u = \nabla u \cdot n$, where n is the normal vector as before. Since $\phi = 0$ on $\partial\Omega$, we obtain

$$-(\Delta u, \phi) = (\nabla u, \nabla \phi)$$

thus

$$(\nabla u, \nabla \phi) = (f, \phi).$$

In the backward direction we show $(V) \to (D)$:

$$(\nabla u, \nabla \phi) = -(\Delta u, \phi) + \int_{\partial\Omega} \partial_n u \phi \, ds.$$

As before $\phi = 0$ on $\partial\Omega$ and we get

$$(\nabla u, \nabla \phi) = -(\Delta u, \phi)$$

thus

$$-(\Delta u, \phi) = (f, \phi).$$

Consequently,

$$(-\Delta u + f, \phi) = 0.$$

Since $-\Delta u + f$ is continuous on Ω we can employ the fundamental lemma of calculus of variations in higher dimensions from which we obtain

$$-\Delta u + f = 0 \quad \forall x \in \Omega.$$

We recover the boundary conditions since we started from $u \in V$, which includes that $u = 0$ on $\partial\Omega$. □

Proposition 8.68 (Fundamental lemma of calculus of variations in $\mathbb{R}^n$)**.** *Let $\Omega \subset \mathbb{R}^n$ be an open domain and let w be a continuous function. Let $\phi \in C_c^\infty(\Omega)$. If*

$$\int_\Omega w(x)\phi(x) \, dx = 0 \quad \forall \phi \in C^\infty(\Omega)$$

then, $w \equiv 0$ in Ω.

Proof. Similar to the 1D version. See also Theorem 6.3.2, p. 314 in [36]. □

Remark 8.69. *We have not yet shown well-posedness of (D) and (V), which is the topic of the next section. Only we have shown so far that if solutions to (D) and (V) exist, then these solutions are equivalent.*

8.10 The Lax-Milgram lemma

We present in this section a result that ensures well-posedness (existence, uniqueness, stability) of linear variational problems.

Formulation 8.70 (Abstract model problem)**.** *Let V be a Hilbert space with norm $||\cdot||_V$. Find $u \in V$ such that*

$$a(u,\phi) = l(\phi) \quad \forall \phi \in V.$$

Definition 8.71 (Assumptions)**.** *We suppose:*

1. *$l(\cdot)$ is a bounded linear form:*
$$|l(u)| \leq C\|u\| \quad \text{for all } u \in V.$$
2. *$a(\cdot,\cdot)$ is a bilinear form on $V \times V$ and continuous:*
$$|a(u,v)| \leq \gamma \|u\|_V \|v\|_V, \quad \gamma > 0, \quad \forall u, v \in V.$$
3. *$a(\cdot,\cdot)$ is coercive (or V-elliptic):*
$$a(u,u) \geq \alpha \|u\|_V^2, \quad \alpha > 0, \quad \forall u \in V.$$

Lemma 8.72 (Lax-Milgram for convex sets)**.** *Let W be a closed and convex subset of a Hilbert space V. Let $a(\cdot,\cdot) : V \times V \to \mathbb{R}$ be a continuous, symmetric, V-elliptic bilinear form. Then, for each $l \in V^*$ the variational problem*

$$a(u,\phi) = l(\phi) \quad \forall \phi \in V$$

has a unique solution $u \in W$. Moreover, we have the stability estimate:

$$\|u\| \leq \frac{1}{\alpha}\|l\|_{V^*}.$$

with

$$\|l\|_{V^*} := \sup_{\varphi \neq 0} \frac{|l(\varphi)|}{||\varphi||_V}.$$

Proof. The proof contains typical arguments often used in optimization and calculus of variations. Consequently, we work with the minimization formulation (M) rather than (V) and finally transfer the result to (V) as in the equivalent characterizations. The goal is to construct a sequence of solutions and to pass to the limit. Then we must show that the limit is contained in the space V.

Existence of a minimum

We define $J(\cdot) = \frac{1}{2}a(\cdot,\cdot) - l(\cdot)$, which is possible since $a(\cdot,\cdot)$ is symmetric. We choose (recall the best approximation proofs and the remarks regarding minimizing sequences therein) again a minimizing sequence $(u_n)_{n\in\mathbb{N}}$ in V is characterized by the property

$$\lim_{n\to\infty} J(u_n) = d := \inf_{\varphi \in V} J(\varphi).$$

Hence

$$2J(\varphi) = a(\varphi,\varphi) - 2l(\varphi) \geq \alpha ||\varphi||_V^2 - 2||l||\,||\varphi||_V, \tag{109}$$

where, in the last term, the operator norm $|l(\varphi)| \leq ||l||||\varphi||_V$ is used. In particular, the linear functional l is bounded and consequently $\|l\|_{V*} < \infty$. This shows $d > -\infty$ because

$$(\|\varphi\|_V^2 - 2\|\,\|\varphi\|_V) \to \infty \quad \text{for } \|\phi\| \to \infty.$$

Cauchy property

It holds

$$a(u-\varphi, u-\varphi) + a(u+\varphi, u+\varphi) = 2a(u,u) + 2a(\varphi,\varphi). \tag{110}$$

Our previous work is used in the next estimation. We begin at the V-ellipticity of a:

$$\begin{aligned}\alpha||u_n-u_m||_V^2 &\leq a(u_n-u_m,u_n-u_m)\\ &\overset{(110)}{=} 2a(u_n,u_n)+2a(u_m,u_m)-a(u_n+u_m,u_n+u_m)\\ &= 2a(u_n,u_n)+2a(u_m,u_m)-4a\left(\frac{u_n+u_m}{2},\frac{u_n+u_m}{2}\right).\end{aligned}$$

Adding zero in form of

$$-4l(u_n)-4l(u_m)+4l(u_n)+4l(u_m) = -4l(u_n)-4l(u_m)+8l\left(\frac{u_n+u_m}{2}\right)$$

yields

$$\alpha||u_n-u_m||_V^2 \leq 2a(u_n,u_n)+2a(u_m,u_m)-4a\left(\frac{u_n+u_m}{2},\frac{u_n+u_m}{2}\right) \tag{111}$$

$$-4l(u_n)-4l(u_m)+8l\left(\frac{u_n+u_m}{2}\right) \tag{112}$$

$$= 2a(u_n,u_n)-4l(u_n)+2a(u_m,u_m)-4l(u_m) \tag{113}$$

$$-4a\left(\frac{u_n+u_m}{2},\frac{u_n+u_m}{2}\right)+8l\left(\frac{u_n+u_m}{2}\right) \tag{114}$$

$$\overset{(109)}{=} 4J(u_n)+4J(u_m)-8J\left(\frac{u_n+u_m}{2}\right). \tag{115}$$

Hence $J\left(\frac{u_n+u_m}{2}\right)\geq d$ (since W is assumed convex here) and

$$\lim_{n\to\infty}J(u_n)+\lim_{m\to\infty}J(u_m)=d+d=2d.$$

Therefore in the limit:

$$\alpha||u_n-u_m||_V^2\leq 4d+4d-8d=0,\quad n,m\to\infty,$$

which shows the Cauchy property in V:

$$\alpha||u_n-u_m||_V^2\to 0\quad (n,m\to\infty).$$

Therefore, the limit u exists in V (recall that V is a Hilbert space and therefore complete - each Cauchy sequence converges) and consequently in W since it is closed in V. It holds $\lim_{n\to\infty}u_n=u$. Since $a(\cdot,\cdot)$ and $l(\cdot)$ are linear and bounded (i.e., continuous), the functional $J(\cdot)$ is linear and continuous as well, and it follows:

$$\lim_{n\to\infty}u_n=u\quad\Rightarrow\quad J(u)=\lim_{n\to\infty}J(u_n)=\inf_{v\in W}=d$$

showing that the limit u exists and is an element of W.

Remark 8.73. *Be careful when the functional $J(u)$ is not linear anymore or when we deal with norms; e.g., in optimal control problems. Here, the continuity of $J(\cdot)$ is not sufficient to proof convergence. One needs to go to* ***weak convergence*** *and deeper results in functional analysis. A nice idea for quadratic optimization problems in Hilbert spaces recapitulating the basic ingredients of functional analysis is provided in [131][Section 2.4 and Section 2.5].*

(M) YIELDS (V)

It remains to show that the minimum of (M) is equivalent to the solution of (V). The following strategy for the minimizing problem is used:

$$J(u)=\min_{\varphi\in V}J(\varphi)\quad\Leftrightarrow\quad J(u)\leq J(u+\varepsilon\varphi),\quad \varepsilon\in\mathbb{R}. \tag{116}$$

A minimum is achieved by differentiation with respect to ε and setting the resulting equation to zero:

$$\frac{d}{d\varepsilon} J(u+\varepsilon\varphi)\Big|_{\varepsilon=0} = 0. \tag{117}$$

The assertion is shown with the help of the equivalence of the variational problem (V) and the previous minimization formulation (116). Thus,

$$a(u,\varphi) = l(\varphi) \quad \forall \varphi \in V. \tag{118}$$

UNIQUENESS.
Consider two solutions $u_1, u_2 \in V$ of the variational equation (118) which imply

$$a(u_1,\varphi) = l(\varphi) \quad \text{and} \quad a(u_2,\varphi) = l(\varphi) \quad \text{for all } \varphi \in V.$$

Subtraction yields

$$a(u_1 - u_2, \varphi) = 0 \quad \text{for all } \varphi \in V.$$

Choosing $\varphi = u_1 - u_2$ gives us

$$a(u_1 - u_2, u_1 - u_2) = \|u_1 - u_2\|_V^2$$

and therefore

$$\alpha \|u_1 - u_2\|_V^2 = 0 \quad \Leftrightarrow \quad u_1 - u_2 = 0 \quad \Leftrightarrow \quad u_1 = u_2.$$

STABILITY.
Follows from consideration of

$$\alpha ||u||_V^2 \le a(u,u) = l(u) \le C||u||_V.$$

Hence

$$||u||_V \le \frac{C}{\alpha} \qquad \text{with } C = \|l\|_{V^*} = \sup_{u\neq 0} \frac{|l(u)|}{||u||_V}.$$

All assertions were shown. □

Remark 8.74. *For a symmetric bilinear form, i.e.,*

$$a(u,\phi) = a(\phi,u)$$

*the **Riesz representation Theorem** 8.75 can be applied. The key observation is that in this case the mapping $u \mapsto \sqrt{a(u,u)}$ defines a scalar product; and consequently a norm because we work in a Hilbert space. Then, there exists a unique element $u \in V$ such that*

$$l(\phi) = a(u,\phi) \quad \forall \phi \in V.$$

The Lax-Milgram lemma is in particular useful for non-symmetric bilinear forms.

Theorem 8.75 (Riesz representation theorem)**.** *Let $\hat{l} : V \to \mathbb{R}$ be a continuous linear functional on a Hilbert space V. Then there exists a unique element $w \in V$ such that*

$$(w,\phi)_V = \hat{l}(\phi) \quad \forall \phi \in V.$$

Moreover, this operation is an isometry, that is to say, $\|\hat{l}\|_* = \|w\|_V$.

Proof. For a proof the Riesz representation theorem, we refer to [27]. □

Proposition 8.76. *In the symmetric case, $a(u,\phi) = (\nabla u, \nabla\phi)$ defines a scalar product on the Hilbert space V. Moreover, the right hand side functional $l(\cdot)$ is linear and continuous. Thus, the assumptions of the Riesz representation theorem are fulfilled and there exists a unique solution to the Poisson problem.*

Exercise 8. *Show that the assumptions of the Lax-Milgram lemma are verified for the 1D Poisson model problem. Thus, the variational formulation is well-posed (existence, uniqueness and stability of a solution).*

Lemma 8.77 (Lax-Milgram - general version)**.** *Let $a(\cdot,\cdot) : V \times V \to \mathbb{R}$ be a continuous, V-elliptic bilinear form. Then, for each $l \in V^*$ the variational problem*

$$a(u,\phi) = l(\phi) \quad \forall \phi \in V$$

has a unique solution $u \in V$. Moreover, we have the stability estimate:

$$\|u\| \leq \frac{1}{\alpha}\|l\|_{V^*}.$$

with

$$\|l\|_{V^*} := \sup_{\varphi \neq 0} \frac{|l(\varphi)|}{||\varphi||_V}.$$

Proof. We recall upfront from before: Riesz: for all $l \in V^*$ there exists $u \in V$ such that

$$(u,\varphi) = l(\varphi) \quad \forall \varphi \in V.$$

We work as follows.
Step 1 (Existence of an operator A)
For all fixed $u \in V$ the mapping $v \mapsto a(u,v)$ is a bounded linear functional on V. Then with Riesz: there exist $Au \in V$ and $f \in V$ such that

$$a(u,\varphi) = (Au,\varphi) \quad \forall \varphi \in V \tag{119}$$
$$l(\varphi) = (f,\varphi) \quad \forall \varphi \in V. \tag{120}$$

Step 2 (Linearity and boundedness of A)
The mapping $u \mapsto Au$ yields a linear bounded operator $A : V \to V$. It holds for the linearity:

$$(A(\lambda_1 u_1 + \lambda_2 u_2),\varphi)_V = a(\lambda_1 u_1 + \lambda_2 u_2,\varphi) = \ldots = (\lambda_1 Au_1 + \lambda_2 Au_2,\varphi)_V$$

This holds for all $\varphi \in V$ and consequently A is linear. The boundedness (i.e., continuity) we obtain as follows:

$$\begin{aligned} \|Au\|_V^2 = (Au,Au) = a(u,Au) \leq \gamma\|u\|\|Au\| \\ \Rightarrow \|Au\|_V \leq \gamma\|u\| \quad \forall u \in V, \end{aligned}$$

showing that A is bounded. For such operators in FA, we the notation $A \in L(V,V)$ is used. To sum-up

$$a(u,\varphi) = l(\varphi) \quad \text{(Variational form)} \tag{121}$$
$$Au = f \quad \text{(Operator equation)} \tag{122}$$

Step 3 (Existence of a fixed point)
We show that a fixed point exists. Specifically, we establish that

$$v \in V \mapsto T_\delta v := v - \delta(Av - f) \in V$$

with $\delta > 0$, is a contraction on V. Then, it holds

$$T_\delta v = v$$

has a unique solution $u \in V$. This solution is then also solution of (121) and (122) because of

$$\begin{aligned} T_\delta v - v = 0 \\ \Leftrightarrow \delta(Av - f) = 0 \\ \Leftrightarrow Av - f = 0 \\ \Leftrightarrow Av = f. \end{aligned}$$

The contradiction now follows from

$$v - \delta(Av - f) = v - \delta Av - \delta f$$

The right term is constant with respect to v and does therefore not play a role in the contraction proof (recall Banach's fixed point theorem in Numerik 1 for iterative schemes for solving $Ax = b$). Then:

$$\|v - \delta Av\|_V^2 = \|v\|^2 - 2\delta a(v, v) + \delta^2 \|Av\|_V^2$$

We now use the coercivity of $a(v, v)$. Then:

$$\|v - \delta Av\|_V^2 \leq \|v\|^2 - 2\delta\alpha\|v\|_V^2 + \delta^2\gamma^2\|v\|_V^2 = (1 - 2\delta\alpha + \delta^2\gamma^2)\|v\|_V^2$$

For the choice of

$$0 < \delta < \frac{2\alpha}{\gamma^2}$$

we have a contraction and thus a fixed point. Due to the equivalences of the formulations in this proof (weak form, operator form, fixed point form), we have obtained a unique solution of

$$u \in V : a(u, \varphi) = l(\varphi) \quad \varphi \in V.$$

Step 4
Uniqueness and the stability are the same as in the previous first proof of Lax-Milgram's lemma. □

8.10.1 The energy norm

As we have seen before, the (symmetric) bilinear form defines a scalar product from which we can define another norm. The continuity and coercivity of the bilinear form yield the **energy norm**:

$$\|v\|_a^2 := a(v, v), \quad v \in V.$$

This norm is aquivalent to the V-norm of the space V, i.e.,

$$c\|v\|_V \leq \|v\|_a \leq C\|v\|_V, \quad \forall v \in V$$

and two positive constants c and C. We can even precisely determine these two constants:

$$\alpha||u||_V^2 \leq a(u, u) \leq \gamma\|u\|_V^2$$

yielding $c = \sqrt{\alpha}$ and $C = \sqrt{\gamma}$. The corresponding scalar product is defined by

$$(v, w)_a := a(v, w).$$

Finally, the best approximation property can be written in terms of the a-norm:

$$(u - u_h, v)_a = 0 \quad \forall v \in V_h.$$

8.10.2 The Poincaré inequality

In order to apply the Lax-Milgram lemma to partial differential equations complemented with homogeneous boundary conditions, we need another essential tool, the so-called **Poincaré inequality**.

Proposition 8.78. *Let $\Omega \subset \mathbb{R}^n$ be a bounded, open, domain. There exists a constant d_Ω (depending on the domain) such that*

$$\int_\Omega |v(x)|^2 \, dx \leq d_\Omega \int_\Omega |\nabla v(x)|^2 \, dx.$$

holds true for each $v \in H_0^1(\Omega)$.

Remark 8.79. *It is clear that the Poincaré inequality specifically holds true for functions from H_0^1 but not H^1. A counter example is $v \equiv 1$, which is in H^1, but violates the Poincaré inequality.*

Proof. The proof can be found in all textbooks, e.g., Rannacher [106], Wloka [145] or Braess [24], page 29. □

Corollary 8.80. *Let $\Omega \subset \mathbb{R}^n$ be an open, bounded, domain. The semi-norm*

$$|v|_{H_0^1} = \Big(\int_\Omega |\nabla v|^2 \, dx \Big)^{1/2}$$

is a norm on H_0^1 and equivalent to the usual H^1 norm.

Proof. Let $v \in H_0^1$. It holds on the one hand (trivial!):

$$|v|_{H_0^1} \leq \|v\|_{H_0^1}.$$

On the other hand, using Poincaré's inequality, we obtain:

$$\|v\|_{H^1}^2 \leq (d_\Omega + 1) \int_\Omega |\nabla v|^2 \, dx = (d_\Omega + 1)|v|_{H_0^1}^2,$$

yielding

$$|v|_{H_0^1}^2 \leq \|v\|_{H^1}^2 \leq (d_\Omega + 1)|v|_{H_0^1}^2.$$

□

8.10.3 Trace theorems

We have seen that H^1 functions are not continuous when $n \geq 2$. Consequently, we cannot define pointwise values and secondly, boundaries of measure zero. Thus, it is a priori a bit difficult to define boundary values in Sobolev spaces. We face the question in which sense boundary values can be defined when working with Sobolev spaces.

We have:

Theorem 8.81 (Trace theorem)**.** *Let Ω be an open, boundary domain of class C^1. We define the **trace** $\gamma_0 : H^1 \cap C(\bar{\Omega}) \to L^2(\partial\Omega) \cap C(\partial\bar{\Omega})$ as*

$$v \mapsto \gamma_0(v) = v|_{\partial\Omega}.$$

By continuity γ_0 can be extended to the entire domain Ω. It holds

$$\|v\|_{L^2(\partial\Omega)} \leq C\|v\|_{H^1(\Omega)}.$$

Proof. See Rannacher [106], page 30. □

Theorem 8.82 (Trace theorem for derivatives)**.** *Let Ω be an open, boundary domain of class C^1. We define the **trace** $\gamma_0 : H^2 \cap C^1(\bar{\Omega}) \to L^2(\partial\Omega) \cap C(\partial\bar{\Omega})$ as*

$$v \mapsto \gamma_1(v) = \partial_n v|_{\partial\Omega}.$$

By continuity, γ_1 can be extended from $H^2(\Omega)$ mapped into $L^2(\partial\Omega)$. It holds

$$\|\partial_n v\|_{L^2(\partial\Omega)} \leq C\|v\|_{H^2(\Omega)}.$$

Proof. The existence of γ_1 follows from the first trace theorem previously shown. Since $v \in H^2$, we have $\nabla v \in (H^1)^n$. Consequently, we can define the trace of ∇v on $\partial\Omega$, which is then a trace on $(L^2(\partial\Omega))^n$. Since the normal vector is a bounded, continuous function on $\partial\Omega$, we have $\nabla v \cdot n \in L^2(\partial\Omega)$. □

Remark 8.83. *The second trace theorem has important consequences on the regularity of the boundary functions. In general, for variational problems we can only proof that the solution u is from H^1. In this case, the normal derivative is only from H^{-1} (a space with very low regularity). Only under additional assumptions on the data and the domain, we can show $u \in H^2$, which would yield better regularity of the normal derivative $\nabla u \cdot n$ on $\partial\Omega$.*

8.11 Theory of elliptic problems (in higher dimensions)

Gathering all results from the previous sections, we are now in a state to proof well-posedness of the Poisson problem in higher dimensions.

8.11.1 The formal procedure

In most cases, one starts with the differential problem (D). In order to establish well-posedness, one can work in three steps:

- Derive formally a variational problem (V).
- Proof well-posedness for the variational problem (here for linear problems using Lax-Milgram).
- If the solution enjoys enough regularity, go back and show $(V) \to (D)$ (see also Section 8.1.2) using the fundamental lemma of calculus of variations.

8.11.2 Poisson's problem: homogeneous Dirichlet conditions

Proposition 8.84. *Let Ω be a bounded domain of class C^1 and let $f \in L^2$. Let $V := H_0^1$. Then, the Poisson problem has a unique solution $u \in V$ and it exists a constant c_p (independant of f) such that the stability estimate*

$$\|u\|_{H^1} \leq c_p \|f\|_{L^2}$$

holds true.

Proof. We use the Lax-Milgram lemma and check its assumptions. Since H_0^1 is a subspace of H^1, we work with the H^1 norm.

Continuity of $a(\cdot,\cdot)$

Using Cauchy-Schwarz, we obtain:

$$\begin{aligned} |a(u,\phi)| = |\int_\Omega \nabla u \cdot \nabla \phi \, dx| &\leq \int_\Omega |\nabla u| \, |\nabla \phi| \, dx \leq \Big(\int_\Omega |\nabla u|^2 \, dx\Big)^{1/2} \Big(\int_\Omega |\nabla \phi|^2 \, dx\Big)^{1/2} \\ &\leq \Big(\int_\Omega (u^2 + |\nabla u|^2) \, dx\Big)^{1/2} \Big(\int_\Omega (\phi^2 + |\nabla \phi|^2) \, dx\Big)^{1/2} = \|u\|_{H^1} \|\phi\|_{H^1}. \end{aligned}$$

Coercivity of $a(\cdot,\cdot)$

Using the Poincaré inequality, we obtain

$$\begin{aligned} a(u,u) = \int_\Omega |\nabla u|^2 \, dx &= \frac{1}{2} \int_\Omega |\nabla u|^2 \, dx + \frac{1}{2} \int_\Omega |\nabla u|^2 \, dx \\ &\geq \frac{1}{2} \int_\Omega |\nabla u|^2 \, dx + \frac{1}{2d_\Omega} \int_\Omega u^2 \, dx \geq \frac{1}{2} \min(1, \frac{1}{d_\Omega}) \|u\|_{H^1}^2. \end{aligned}$$

Continuity of $l(\cdot)$

Using again Cauchy-Schwarz, we obtain:

$$|l(\phi)| = |(f,\phi)_{L^2}| \leq \|f\|_{L^2} \|\phi\|_{L^2} \leq \|f\|_{L^2} \|\phi\|_{H^1}.$$

Stability estimate

From Lax-Milgram we know:

$$\|u\| \leq \frac{1}{\alpha} \|l\|_{V^*} \leq \frac{|l(\phi)|}{||\phi||_V} \leq \frac{\|f\|_{L^2} \|\phi\|_{H^1}}{\|\phi\|_{H^1}} = \frac{1}{\alpha} \|f\|_{L^2}.$$

Everything is shown now.

□

Remark 8.85 (On the dual norm)**.** *Previously we have had:*

$$|l(\phi)| = |(f,\phi)| \leq \|f\|_{L^2}\|\phi\|_{L^2}.$$

When $f \in L^2$. But this estimate works also for $f \in H^{-1}$ the dual space of H^1. Then:

$$|l(\phi)| = |(f,\phi)| \leq \|f\|_{H^{-1}}\|\phi\|_{H^1}.$$

The smallest possible choice for $C = \|l\|_{V^*} = \|f\|_{H^{-1}}$ (see Section 3.12) is

$$C = \sup_{\phi \in H_0^1, \phi \neq 0} \frac{|(f,\phi)|}{\|\phi\|_{H^1}},$$

which is indeed a norm and in general denoted by $\|f\|_{H^{-1}}$. In the L^2 case we have

$$\|f\|_{L^2} = \sup_{\phi \in L^2, \phi \neq 0} \frac{|(f,\phi)|}{\|\phi\|_{L^2}}.$$

Here, we take the supremum over a larger space; recall that $H^1 \subset L^2$. Consequently

$$\|f\|_{H^{-1}} \leq \|f\|_{L^2}$$

and therefore

$$L^2 \subset H^{-1}$$

and finally

$$H^1 \subset L^2 \subset H^{-1}.$$

Under stronger conditions, we have the following regularity result:

Proposition 8.86 (Regularity of the exact solution)**.** *Let Ω be bounded, open and of class C^2. Then, the solution of the Poisson problem is $u \in H^2(\Omega)$ and it holds the stability estimate:*

$$\|u\|_{H^2} \leq C\|f\|_{L^2}.$$

Proof. See for instance [89]. □

Proposition 8.87 (Higher regularity of the exact solution)**.** *Let Ω be bounded, open and of class C^k (i.e., very smooth). Then, the solution of the Poisson problem is $u \in H^{k+2}(\Omega)$ and it holds the stability estimate:*

$$\|u\|_{H^{k+2}} \leq C\|f\|_{H^k}.$$

If the boundary is not smooth enough (even when f is sufficiently smooth), this result will not hold true anymore. This is the case for reentrent corners (L-shaped domains) for instance.

8.11.3 Nonhomogeneous Dirichlet conditions

In this section, we investigate non-homogeneous Dirichlet boundary conditions. Despite one only needs to add formally a value or a function for the boundary conditions, the mathematical formulations are non-trivial.

We have

Formulation 8.88. *Let $\Omega \subset \mathbb{R}^n$ an open and bounded domain. Let $f \in L^2(\Omega)$. Find u such that*

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega, \\ u &= h \quad \text{on } \partial\Omega. \end{aligned}$$

Since we seek $u \in H^1(\Omega)$ we assume that there exists a function $u_0 \in H^1$ such that

$$trace(u_0) = h \quad \text{on } \partial\Omega.$$

We know that $h \in L^2(\partial\Omega)$.

In the following, we concentrate first on the derivation of a weak formulation. The idea is to work with the space H^1 by extracting the nonhomogeneous Dirichlet condition. We write

$$u = u_0 + \tilde{u}, \quad \tilde{u} \in H_0^1.$$

Then:

$$\int_\Omega \nabla u \cdot \nabla \phi \, dx = \int_\Omega \nabla(u_0 + \tilde{u}) \cdot \nabla \phi \, dx = \int_\Omega (\nabla u_0 \cdot \nabla \phi + \tilde{u} \cdot \nabla \phi) \, dx$$

This yields the variational problem:

Formulation 8.89. *Find $\tilde{u} \in H_0^1$ such that*

$$(\nabla \tilde{u}, \nabla \phi) = (f, \phi) - (\nabla u_0, \nabla \phi) \quad \forall \phi \in H_0^1.$$

Having found $\tilde{u}$, we obtain $u \in H^1$ via

$$u = u_0 + \tilde{u}.$$

The equivalent formulation, often found in the literature, is:

Formulation 8.90. *Find $u \in \{h + H_0^1\}$ such that*

$$(\nabla u, \nabla \phi) = (f, \phi) \quad \forall \phi \in H_0^1.$$

Example 8.91 (Poisson in 1D)**.** *Let*

$$-u''(x) = f,$$
$$h = 4 \quad \text{on } \partial\Omega.$$

Here $u_0 = 4$ in Ω. When we solve the Poisson problem with finite elements, we proceed as shown before and obtain $\tilde{u} \in H_0^1$ and finally simply

$$u = \tilde{u} + 4$$

to obtain the final u. For nonconstant u_0 (thus nonconstant h on $\partial\Omega$, please be careful in the variational formulation since the term

$$(\nabla u_0, \nabla \phi)$$

will not vanish and needs to be assembled. For instance let $u(0) = 0$ and $u(1) = 1$ in $\Omega = (0, 1)$. Then, $u_0 = x$ and $trace(u_0) = h$ with $h(0) = 1$ and $h(1) = 1$. Thus we solve Formulation 8.89 to obtain $\tilde{u}$ and finally:

$$u = \tilde{u} + u_0 = \tilde{u} + x.$$

Proposition 8.92. *Formulation 8.89 yields a unique solution.*

Proof. The proof is similar to Formulation 8.84. □

8.11.4 Neumann conditions

We have seen in the 1D case that the pure Neumann problem has solutions defined up to a constant value.

Formulation 8.93 (Pure Neumann problem)**.** *Let $\Omega \subset \mathbb{R}^n$ be an open, bounded, connected, domain and $f \in L^2(\Omega)$ and $g \in L^2(\partial\Omega)$. We consider*

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

Remark 8.94. *We call the problem 'pure' Neumann problem because it is of course possible to have different boundary conditions on different boundary parts. But once we have Dirichlet conditions on some part of the boundary $\partial\Omega_D$ with $\partial\Omega_D \neq \emptyset$ and on the remaining boundary $\partial\Omega_N$ Neumann conditions, the problem of non-uniqueness is gone since the solution is fixed on the Dirichlet part.*

It only exists a solution if the following compatibility condition is fulfilled:

$$\int_\Omega f(x)\,dx + \int_{\partial\Omega} g(x)\,ds = 0. \tag{123}$$

The compatibility condition is necessary and sufficient. We have the following result:

Proposition 8.95. *Let $\Omega \subset \mathbb{R}^n$ an open, bounded, connected domain of class C^1. Let $f \in L^2(\Omega)$ and $g \in L^2(\partial\Omega)$, which satisfy the compabitiblity condition* (123)*. Then, it exists a weak solution $u \in H^1(\Omega)$ of Formulation 8.93, which is unique up to a constant value.*

Proof. The variational formulation is obtained as usually:

$$(\nabla u, \nabla\phi) = (f,\phi) + \langle g,\phi\rangle \quad \text{for all admissible } \phi.$$

Here, we have the first difficulty, since the Hilbert space H^1 will not yield the coercivity of the corresponding bilinear form (in fact this is related to the non-uniqueness). For this reason, we define the space:

$$V := \{v \in H^1 |\ \int_\Omega v\,dx = 0\}.$$

Hence, the correct variational formulation is:

$$\text{Find } u \in V: \quad (\nabla u, \nabla\phi) = (f,\phi) + \langle g,\phi\rangle \quad \forall \phi \in V.$$

As before, we adopt the Lax-Milgram Lemma to show existence and uniqueness of the solution. The continuity of the bilinear form and right hand side are obvious; see the proof of Proposition 8.84.

The only delicate step is the coercivity. Here, Poincaré's inequality will not work and we have to adopt a variant of Friedrichs inequality

$$\|v\|_{L^2} \le c(|\tilde{v}| + |v|_{H^1}) \quad \text{for } v \in H^1$$

with $c := c(\Omega)$ and with $\tilde{v} = \int_\Omega v\,dx/\mu(\Omega)$ being the mean value of v. Thus, it holds:

$$a(u,u) = \int_\Omega |\nabla u|^2\,dx = |u|^2_{H^1} \ge \alpha \|u\|^2_{H_1}.$$

Thus, the Lax-Milgram lemma yields a unique solution in the space V. We notice that this solution must satisfy the compatibility condition (123) because Gauss' divergence theorem yields

$$\int_\Omega f\,dx = -\int_\Omega div(\nabla u) = -\int_{\partial\Omega} \nabla u \cdot n\,ds = -\int_{\partial\Omega} g\,ds.$$

Hence

$$\int_\Omega f\,dx + \int_{\partial\Omega} g\,ds = 0.$$

Interestingly this condition is also sufficient. Lax-Milgram says that we obtain the solution $u \in V$ via solving

$$a(u,\phi) = (f,\phi) + \langle g,\phi\rangle \quad \forall \phi \in V.$$

Using the compatibility condition we see the following:

$$\begin{aligned}
&\int_\Omega f\,dx + \int_{\partial\Omega} g\,ds = 0,\\
\Leftrightarrow\quad & c\int_\Omega f\,dx + c\int_{\partial\Omega} g\,ds = 0\\
\Leftrightarrow\quad & \int_\Omega f\cdot c\,dx + \int_{\partial\Omega} g\cdot c\,ds = 0\\
\Leftrightarrow\quad & \int_\Omega f\cdot \phi\,dx + \int_{\partial\Omega} g\cdot \phi\,ds = 0\\
\Leftrightarrow\quad & a(u,\phi) = (f,\phi) + \langle g,\phi\rangle.
\end{aligned}$$

We observe that this relation holds in particular for $\phi = const$. For more information see Braess [24]. □

8.11.5 Robin conditions

Given:

Formulation 8.96. *Let $\Omega \subset \mathbb{R}^n$ and $\partial\Omega$ its boundary. Furthermore, $f \in L^2(\Omega)$ and $g \in L^2(\partial\Omega)$ and $c \in L^\infty(\Omega)$ and $b \in L^\infty(\partial\Omega)$. Then: Find u such that*

$$\begin{aligned} -\Delta u + cu &= f, \quad \text{in } \Omega, \\ \partial_n u + bu &= g, \quad \text{on } \partial\Omega. \end{aligned}$$

We first formulate a weak formulation. As function space, we use $V := H^1$. The bilinear and linear forms are given by:

$$\begin{aligned} a(u,\phi) &= \int_\Omega \nabla u \cdot \nabla \phi \, dx + \int_\Omega cu\phi \, dx + \int_{\partial\Omega} bu\phi \, ds, \\ l(\phi) &= \int_\Omega f\phi \, dx + \int_{\partial\Omega} g\phi \, ds. \end{aligned}$$

We notice that the functional l is not anymore a L^2 function, but only a linear functional on the space V.

To proof coercivity, we need the following general form of Friedrichs inequality:

Lemma 8.97 (Friedrichs inequality)**.** *Let $\Omega \subset \mathbb{R}^n$ a bounded domain of class C^1 and $\Gamma \subset \partial\Omega$ a measurable set with $|\Gamma| > 0$. Then there exists a constant C (independent of u) such that*

$$\|u\|_{H^1}^2 \leq C\Big(|u|_{H^1}^2 + \Big(\int_\Gamma u \, ds\Big)^2\Big) \quad \forall y \in H^1(\Omega).$$

Proof. For a proof see [145]. □

Proposition 8.98. *Let Ω be a bounded domain of class C^1 and functions $c \in L^\infty(\Omega)$ and $b \in L^\infty(\partial\Omega)$ almost everywhere non-negative with the condition:*

$$\int_\Omega c^2 \, dx + \int_{\partial\Omega} b^2 \, ds > 0.$$

Then, the variational Robin problem has for each $f \in L^2, g \in L^2(\partial\Omega)$ a unique solution $u \in H^1 =: V$. Furthermore, it holds the stability estimate:

$$\|u\|_{H^1} \leq C\big(\|f\|_{L^2} + \|g\|_{L^2(\partial\Omega)}\big).$$

Proof. We verify again the assumptions of the Lax-Milgram lemma. We work with the Cauchy-Scharz inequality, the generalization of Friedrichs inequality and also the trace lemma since we have deal with functions on the boundary $\partial\Omega$.

Preleminaries

First:

$$|\int_\Omega cu\phi \, dx| \leq \|c\|_{L^\infty}\|u\|_{L^2}\|\phi\|_{L^2} \leq \|c\|_{L^\infty}\|u\|_{H^1}\|\phi\|_{H^1}.$$

With the trace lemma, we have:

$$|\int_{\partial\Omega} bu\phi \, dx| \leq \|b\|_{L^\infty(\partial\Omega)}\|u\|_{L^2(\partial\Omega)}\|\phi\|_{L^2(\partial\Omega)} \leq C\|b\|_{L^\infty(\partial\Omega)}\|u\|_{H^1(\partial\Omega)}\|\phi\|_{H^1(\partial\Omega)}.$$

Continuity of $a(\cdot,\cdot)$

We establish the continuity:

$$|a(u,\phi)| = |(\nabla u, \nabla\phi) + (cu,\phi)_{L^2} + (bu,\phi)_{L^2(\partial\Omega)}| \leq C\|u\|_{H^1}\|\phi\|_{H^1}.$$

In the last estimates we used the inequalities derived in the preleminaries.

Coercivity of $a(\cdot,\cdot)$

$$a(u,u) \geq \frac{\min(1,C_1)}{C_2\max(1,|\Gamma|)}\|u\|^2_{H^1}$$

where $b(x) \geq C_1$. We left out several steps, which can be verified by the reader.

Continuity of $l(\phi)$

We estimate:

$$|l(\phi)| = |\int_\Omega f\phi\,dx + \int_{\partial\Omega} g\phi\,ds| \leq C(\|f\|_{L^2} + \|g\|_{L^2(\partial\Omega)})\|\phi\|_{H^1}.$$

Stability estimate

From Lax-Milgram we know:

$$\|u\|_{H^1} \leq \frac{1}{\alpha}\|l\|_{V^*} \leq \frac{|l(\phi)|}{||\phi||_V} \leq \frac{C(\|f\|_{L^2} + \|g\|_{L^2})\|\phi\|_{H^1}}{\|\phi\|_{H^1}} = C(\|f\|_{L^2} + \|g\|_{L^2}).$$

Everything is shown now.

□

8.12 Finite element spaces

We have now a theoretical basis of the finite element method and for 1D problems we explained in Section 8.4 how to construct explicitely finite elements and how to implement them in a code.

It remains to give a flavour of the construction of finite elements in higher dimensions. Since the major developments have been taken place some time ago, there are nice books including all details:

- Johnson [82] provides an elementary introduction that is easily accessible.
- The most classical book is by Ciarlet [35].
- Another book including many mathematical aspects is by Rannacher [106].

Consequently, the aim of this chapter is only to provide the key ideas to get the reader started. For further studies, we refer to the literature. In addition, to the three above references the reader may consult the books mentiond in Chapter 1 of these lecture notes.

We follow chapter 3 in [82] to provide an elementary introduction. The generalization is nicely performed by Ciarlet [35][Chapter 2]. Moreover, Ciarlet also introduces in details rectangular finite elements and parametric and isoparametric finite elements in order to approximate curved boundaries. Isoparametric elements are also explained in detail by Rannacher [106].

Finite element spaces consist of piecewise polynomials functions on a decomposition (historically still called triangulation), i.e., a mesh,

$$\mathcal{T}_h := \bigcup_i K_i$$

of a bounded domain $\mathbb{R}^n, n = 1,2,3$. The K_i are called **mesh elements**.

In particular:

- $n = 1$: The elements K_i are intervals;

- $n = 2$: The elements K_i are triangles or quadrilaterals;
- $n = 3$: The elements K_i are tetrahedrons or hexahedra.

A **conforming** FEM scheme requires $V_h \subset V$ with $V := H^1$ for instance. Since V_h consists of piecewise polynomials, we have

$$V_h \subset H^1(\Omega) \quad \Leftrightarrow \quad V_h \subset C^0(\bar{\Omega}).$$

As we have seen in the 1D case, to specify a **finite element** (see Section 8.4.7 - we remind that this section holds not only for 1D, but also higher dimensions), we need three ingrediants:

- A mesh $\mathcal{T}_h$ representing the domain Ω.
- Specification of $v \in V_h$ on each element K_i.
- Parameters (degrees of freedom; DoFs) to describe v uniquely on K_i.

8.12.1 Example: a triangle in 2D

Let us follow [82][Chapter 3] and let us work in $2D$. Let K be an element of $\mathcal{T}_h$. A $P_1(K)$ function is defined as

$$v(x) = a_{00} + a_{10}x_1 + a_{01}x_2, \quad x = (x_1, x_2) \in K$$

and coefficients $a_{ij} \in \mathbb{R}$.

Definition 8.99 (A basis of P_1 (linear polynomials))**.** *A basis of the space P_1 is given by*

$$P_1 = \{\phi_1, \phi_2, \phi_3\}$$

with the basis functions

$$\phi_1 \equiv 1, \quad \phi_2 = x_1, \quad \phi_3 = x_2.$$

The dimension is $dim(P_1) = 3$. Clearly, any polynomial from P_1 can be represented through the linear combination:

$$p(x) = a_{00}\phi_1 + a_{10}\phi_2 + a_{01}\phi_3.$$

Definition 8.100 (A basis of P_2 (quadratic polynomials))**.** *A basis of the space P_1 is given by*

$$P_2 = \{\phi_1, \phi_2, \phi_3, \phi_4, \phi_5, \phi_6\}$$

with the basis functions

$$\phi_1 \equiv 1, \quad \phi_2 = x_1, \quad \phi_3 = x_2, \quad \phi_4 = x_1^2, \quad \phi_5 = x_2^2, \quad \phi_6 = x_1x_2.$$

The dimension is $dim(P_1) = 6$. Again, any polynomial from P_2 can be represented through the linear combination:

$$p(x) = a_{00}\phi_1 + a_{10}\phi_2 + a_{01}\phi_3 + a_{20}\phi_4 + a_{02}\phi_5 + a_{11}\phi_6.$$

Definition 8.101 (A basis of P_s)**.** *A basis of the space P_1 is given by*

$$P_s = \{v|\ v(x) = \sum_{0 \le i+j \le s} a_{ij}x_1^i x_2^j\}$$

with the dimension

$$dim(P_s) = \frac{(s+1)(s+2)}{2}.$$

In the following, we explicitely construct the space of linear functions on K. Let

$$V_h = \{v \in C^0(\bar{\Omega})|\ v|_K \in P_1(K), \quad \forall K \in \mathcal{T}_h\},$$

which we still know from our 1D constructions. To uniquely describe the functions in V_h we choose the global degrees of freedom, here the values of the nodal points of $\mathcal{T}_h$.

Remark 8.102. *Choosing the nodal points is the most common choice and is a Lagrangian scheme. Using for instance derivative information in the nodal points, we arrive at Hermite polynomials.*

We need to show that the values of the nodal points yield a unique polynomial in $P_1(K)$.

Definition 8.103 (Local DoFs)**.** *Let $n = 2$ and let K be a triangle with vertices $a^i, i = 1, 2, 3$. These values are called* ***local degrees of freedom****.*

Proposition 8.104. *Let $K \in \mathcal{T}_h$ be an element (a triangle) with vertices $a^i = (a_1^i, a_2^i) \in K$. A function on K is uniquely determined by the local degrees of freedom from Definition 8.103. In other words, for given values α_i, there exists a unique function $v \in P_1(K)$ such that*

$$v(a^i) = \alpha_i, \quad i = 1, 2, 3.$$

Proof. Recalling the definition of a P_1 polynomial on K, we have

$$v(x) = a_{00} + a_{10}x_1 + a_{01}x_2.$$

In the support points a^i, we have:

$$v(a^i) = a_{00} + a_{10}a_1^i + a_{01}a_2^i = \alpha_i, \quad i = 1, 2, 3.$$

This yields a quadratic system with three equations for three unknowns. Using matrix notation, we can plug the coefficients into a system

$$Ba = b$$

with

$$B = \begin{pmatrix} 1 & a_1^1 & a_2^1 \\ 1 & a_1^2 & a_2^2 \\ 1 & a_1^3 & a_2^3 \end{pmatrix}, \quad a = \begin{pmatrix} a_{00} \\ a_{10} \\ a_{01} \end{pmatrix}, \quad b = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix}.$$

The system $Ba = b$ has a unique solution when $det(B) \neq 0$, which can be easily checked here and completes the proof. □

Algorithm 8.105 (Construction of P_1 shape functions)**.** *For the explicit construction of such P_1 shape function, we simply need to solve*

$$v(a^i) = a_{00} + a_{10}a_1^i + a_{01}a_2^i = \alpha_i, \quad i = 1, 2, 3,$$

for the unknown coefficients a_{ij}. This means, we need to construct three basis functions $\psi_i \in P_1(K)$ with

$$\psi_i(a^j) = \delta_{ij},$$

where δ_{ij} is the already introduced Kronecker delta. Thus, we have to solve the system $Ba = b$ three times:

$$\psi_i(a^j) = a_{00} + a_{10}a_1^j + a_{01}a_2^j = \alpha_j, \quad i, j = 1, 2, 3.$$

Then, we can build the local basis functions via

$$\psi_i(x) = a_{00}^i + a_{10}^i x_1 + a_{01}^i x_2$$

with all quantities explicitely given.

We finally check that the continuity at the nodes a^i is sufficient to obtain a globally continuous function, which is in particular also continuous at the edges between two elements. Let K_1 and K_2 be two elements in $\mathcal{T}_h$ with a common edge Γ and the end points b_1 and b_2. Let $v_i = v|_{K_i} \in P_1(K_i)$ be the restriction of v to K_i. Then, the function $w = v_1 - v_2$ defined on Γ vanishes at the end points b_1 and b_2. Since w is linear on Γ the entire function w must vanish on Γ. Consequently, v is continuous across Γ. Extending to all edges of $\mathcal{T}_h$ be conclude that the function v is a globally continuous function and that indeed $v \in C^0(\bar{\Omega})$.

Definition 8.106 (C^0 elements)**.** *The class of finite elements that fulfills the continuity condition at the edges are called C^0 elements.*

Remark 8.107. *The generalization to n-simplicies can be found in [35][Chapter 2].*

8.12.2 Example: a quadrilateral in 2D

In this section, we briefly present how quadrilaterals can be constructed. The work program of this section is the same as in Section 8.12.1 and is based on [82][Chapter 3].

Let K be an element with four vertices $a^i, i = 1, \dots 4$ and the sides parallel to the coordinate axes in $\mathbb{R}^2$.

Definition 8.108 (A basis of Q_1 (bilinear polynomials))**.** *A basis of the space $Q_1(K)$ on an element K is given by*

$$Q_1 := Q_1(K) = \{\phi_1, \phi_2, \phi_3, \phi_4\}$$

with the basis functions

$$\phi_1 \equiv 1, \quad \phi_2 = x_1, \quad \phi_3 = x_2, \quad \phi_4 = x_1 x_2.$$

The dimension is $dim(P_1) = 3$. Clearly, any polynomial from P_1 can be represented through the linear combination:

$$p(x) = a_{00}\phi_1 + a_{10}\phi_2 + a_{01}\phi_3 + a_{11}\phi_4.$$

Proposition 8.109. *A Q_1 function is uniquely determined by the values of the nodal points a^i.*

Proof. The same as for triangles. □

Proposition 8.110. *The definition of $Q_1(K)$ yields globally continuous functions.*

Proof. Same as for triangles. □

Remark 8.111. *As for triangles, we can easily defined higher-order spaces.*

Remark 8.112. *Be careful, when quadrilateral elements are turned or shifted. Then, the nature of the local polynomials will change. This is one major reason why in practice we work with a master element and all other elements are obtained via transformations from this master element. To this end, we are not restricted to a specific coordinate system and leaves the desired freedom for deformations of the single elements. A lot of details can be found in [106][Pages 105 ff].*

8.12.3 Well-posedness of the discrete problem

With the help of the Lax-Milgram lemma, we obtain:

Proposition 8.113. *Let V a real Hilbert space and $V_h \subset V$. Let $a(u, v)$ be a bilinear form and $l(v)$ a linear form. Then, the discrete problem*

$$\text{Find } u_h \in V_h : a(u_h, \phi_h) = l(\phi_h) \quad \forall \phi_h \in V_h$$

has a unique solution. This discrete solution $u_h \in V_h$ is obtained by solving a linear equation system.

Proof. Existence and uniqueness are obtained from the Lax-Milgram lemma with the same arguments as for the continuous problem. In particular, we have

$$AU = b$$

with

$$(A)_{ij=1}^n = a(\phi_j, \phi_i), \quad (b)_{i=1}^n = l(\phi_i).$$

The coercivity yields the positive definitness of the matrix A and therefore (see lectures on linear algebra), A is invertible:

$$AU \cdot U \geq \alpha \| \sum_{j=1}^n u_j \phi_j \| \geq C \|U\|^2.$$

In fact, we investigate the homogeneous system

$$a(u_h, \phi_h) = 0$$

and show that this only yields the zero solution, which is immediately true using the coercivity.

Furthermore, the symmetry of $a(u, \phi)$ yields the symmetry of A. Linear algebra arguments on matrices (symmetric, positive, definite) yield a unique solution U. □

8.13 Numerical analysis: error estimates

In this section, we perform the numerical analysis of our finite element derivations. We work as before and focus on elementary aspects working out 1D results in order to show the mechanism and to explain the principle ideas. For higher dimensions, we only provide the results and refer to the literature for detailed proofs.

8.13.1 The Céa lemma

The Céa lemma is the only result in this section that holds for higher dimensions. We investigate the best approximation error (recall our 1D findings).

Proposition 8.114 (Céa lemma)**.** *Let V be a Hilbert space and $V_h \subset V$ a finite dimensional subspace. Let the assumptions, Def. 8.71, of the Lax-Milgram Lemma 8.72 hold true. Let $u \in V$ and $u_h \in V_h$ be the solutions of the variational problems. Then:*

$$\|u - u_h\|_V \leq \frac{\gamma}{\alpha} \inf_{\phi_h \in V_h} \|u - \phi_h\|_V$$

Proof. It holds Galerkin orthogonality:

$$a(u - u_h, w_h) = 0 \quad \forall w_h \in V_h.$$

We choose $w_h := u_h - \phi_h$ and we obtain:

$$\alpha\|u - u_h\|^2 \leq a(u - u_h, u - u_h) = a(u - u_h, u - \phi_h) \leq \gamma\|u - u_h\|\|u - \phi_h\|.$$

This yields

$$\|u - u_h\| \leq \frac{\gamma}{\alpha}\|u - \phi_h\| \quad \forall \phi_h \in V_h.$$

Taking the infimum yields:

$$\|u - u_h\| \leq \inf_{\phi_h \in V_h} \frac{\gamma}{\alpha}\|u - \phi_h\|.$$

Since V_h is closed it even holds

$$\|u - u_h\| \leq \min_{\phi_h \in V_h} \frac{\gamma}{\alpha}\|u - \phi_h\|.$$

□

Corollary 8.115. *When $a(\cdot, \cdot)$ is symmetric, which is the case for Poisson, then the improved estimate holds true:*

$$\|u - u_h\|_V = \sqrt{\frac{\gamma}{\alpha}} \inf_{\phi_h \in V_h} \|u - \phi_h\|_V$$

Based on the Céa lemma, we have the first (still non-quantitative) error estimate. The key aspect is that we work with a dense subspace U of V which contains simpler functions than V itself from which we can interpolate from U to the discrete space V_h. We have [3]:

Proposition 8.116. *We assume that the hypotheses from before hold true. Furthermore, we assume that $U \subset V$ is dense. We construct an interpolation operator $i_h : U \to V_h$ such that*

$$\lim_{h \to 0} \|v - i_h(v)\| = 0 \quad \forall v \in U$$

holds true. Then, for all $u \in V$ and $u_h \in V_h$:

$$\lim_{h \to 0} \|u - u_h\| = 0.$$

This result shows that the Galerkin solution $u_h \in V_h$ converges to the continuous solution u.

Proof. Let $\varepsilon > 0$. Thanks to the density, for each $u \in V$, it exists a $v \in U$ such that $\|u - v\| \leq \varepsilon$. Moreover, there is an $h_0 > 0$, depending on the choice of ε, such that

$$\|v - i_h(v)\| \leq \varepsilon \quad \forall h \leq h_0.$$

The Céa lemma yields now:

$$\|u - u_h\| \leq C\|u - i_h(v)\| = C(\|u - v + v - i_h(v)\|) \leq C(\|u - v\| + \|v - i_h(v)\|) \leq C(\varepsilon + \varepsilon) = 2C\varepsilon.$$

These proofs employs the trick, very often seen for similar calculations, that at an appropriate place an appropriate function, here v is inserted, and the terms are split thanks to the triangular inequality. Afterwards, the two separate terms can be estimated using the assumptions of other known results. □

8.13.2 H^1 and L^2 estimates in 1D

First, we need to construct an interpolation operator in order to approximate the continuous solution at certain nodes.

Definition 8.117. *Let $\Omega = (0, 1)$. Let $V_h = \{\varphi_0, \ldots, \varphi_{n+1}\}$ a discrete function space. Moreover, let $v \in H^1$. A P_1 interpolation operator $i_h : H^1 \to V_h$ is defined by*

$$(i_h v)(x) = \sum_{j=0}^{n+1} v(x_j)\varphi_j(x).$$

Recapitulate for instance Lagrangian interpolation, e.g., [114][Chapter 8.1].

This definition is well-defined since H^1 functions are continuous in 1D and are pointwise defined. The interpolation i_h creates a piece-wise linear function that coincides in the support points x_j with its H^1 function.

Remark 8.118. *Since H^1 functions are not continuous any more in higher dimensions, we need more assumptions here.*

The convergence of a finite element method in 1D relies on

Lemma 8.119. *Let $i_h : H^1 \to V_h$ be given. Then:*

$$\lim_{h \to 0} \|u - i_h u\|_{H^1} = 0.$$

If $u \in H^2$, there is a constant C such that

$$\|u - i_h u\|_{H^1} \leq Ch|u|_{H^2}.$$

Proof. Since

$$\|u - i_h u\|_{H^1}^2 = \|u - i_h u\|_{L^2}^2 + |u - i_h u|_{H^1}^2,$$

the result follows immediately from the next two lemmas; namely Lemma 8.121 and Lemma 8.122. □

Definition 8.120. *The global norm $\|\cdot\|$ is obtained by locally integrating and then summing up the results on all elements. For instance, the L^2 norm is defined by*

$$\|u\|_{L^2} = \Big(\sum_{K \in T_h} \int_K |u|^2 \, dx \Big)^{1/2}.$$

Lemma 8.121. *For a function $u \in H^2$, it exists a constant C (independent of h) such that*

$$\|u - i_h u\|_{L^2} \leq Ch^2 \|u''\|_{L^2},$$
$$|u - i_h u|_{H^1} \leq Ch \|u''\|_{L^2}.$$

Proof. We choose $u \in C_c^\infty(\Omega)$ (the result will hold true for H^2 because of a density argument). The interpolant $i_h u$ is a linear function on each element $K_j = (x_j, x_{j+1})$. We calculate:

$$\begin{aligned} u(x) - i_h u(x) &= u(x) - \Big(u(x_j) + \frac{u(x_{j+1}) - u(x_j)}{x_{j+1} - x_j}(x - x_j)\Big) \\ &= u(x) - u(x_j) - \frac{x - x_j}{x_{j+1} - x_j}\Big(u(x_{j+1}) - u(x_j)\Big) \\ &= \int_{x_j}^{x} u'(t)\,dt - \frac{x - x_j}{x_{j+1} - x_j}\int_{x_j}^{x_{j+1}} u'(t)\,dt. \end{aligned}$$

Since u' is continuous, we can apply the mean value theorem from calculus (see e.g., [85][Chapter 11.3]) and further obtain:

$$\begin{aligned} u(x) - i_h u(x) &= (x - x_j)u'(x_j + \alpha) - \frac{x - x_j}{x_{j+1} - x_j}(x_{j+1} - x_j)u'(x_j + \beta) \\ &= (x - x_j)\int_{x_j+\beta}^{x_j+\alpha} u''(t)\,dt, \end{aligned}$$

with $0 \le \alpha \le x - x_j$ and $0 \le \beta \le h, h = x_{j+1} - x_j$. We now take the square and apply the Cauchy-Schwarz inequality (recall $|(u,v)|^2 \le \|u\|^2\|v\|^2$):

$$\begin{aligned} |u(x) - i_h u(x)|^2 &= \Big|(x - x_j)\int_{x_j+\beta}^{x_j+\alpha} u''(t)\,dt\Big|^2 \\ &\le h^2\Big|\int_{x_j+\beta}^{x_j+\alpha} u''(t)\,dt\Big|^2 \\ &\le h^2\Big(\int_{x_j+\beta}^{x_j+\alpha} 1^2\,dt\Big)\Big(\int_{x_j+\beta}^{x_j+\alpha} |u''(t)|^2\,dt\Big) \\ &\le h^2\Big(\int_{x_j}^{x_{j+1}} 1^2\,dt\Big)\Big(\int_{x_j}^{x_{j+1}} |u''(t)|^2\,dt\Big) \\ &= h^3\Big(\int_{x_j}^{x_{j+1}} |u''(t)|^2\,dt\Big). \end{aligned}$$

When we now integrate $|u(x) - i_h u(x)|^2$ on K_j, we obtain a norm-like object what we wish:

$$\int_{x_j}^{x_{j+1}} |u(x) - i_h u(x)|^2\,dx \le h^4\Big(\int_{x_j}^{x_{j+1}} |u''(t)|^2\,dt\Big).$$

Summing up over all elements $K_j = [x_j, x_{j+1}], j = 0, \dots, n$, we obtain the global norm (see Definition 8.120):

$$\sum_j \int_{x_j}^{x_{j+1}} |u(x) - i_h u(x)|^2\,dx = \|u - i_h u\|_{L^2}^2 \le h^4 |u|_{H^2}^2.$$

Taking the square root yields the desired result:

$$\|u - i_h u\|_{L^2} \le Ch^2\|u''\|_{L^2} \quad \text{for } u \in C_c^\infty(\Omega).$$

By density the result also holds true for H^2.

The second estimate is obtained by the same calculations. Here, the initial idea is:

$$\begin{aligned} u'(x) - i_h u'(x) &= u'(x) - \frac{u(x_{j+1}) - u(x_j)}{h} \\ &= \frac{hu'(x)}{h} - \frac{1}{h}\int_{x_j}^{x_{j+1}} u'(t)\,dt \\ &= \frac{1}{h}\int_{x_j}^{x_{j+1}} (u'(x) - u'(t))\,dt \\ &= \frac{1}{h}\int_{x_j}^{x_{j+1}} \int_{t}^{x} u''(s)\,ds. \end{aligned}$$

As before, taking the square root, then using Cauchy-Schwarz, then integrating over each element K_j, and finally summing over all elements K_j will yield the desired result

$$|u - i_h u|_{H^1} \leq Ch\|u''\|_{L^2}.$$

□

Lemma 8.122. *There exists a constant C (independent of h) such that for all $u \in H^1(\Omega)$, it holds*

$$\|i_h u\|_{H^1} \leq C\|u\|_{H^1}$$

and

$$\|u - i_h u\|_{L^2} \leq Ch|u|_{H^1}.$$

Moreover:

$$\lim_{n\to\infty} \|u' - i_h u'\|_{L^2} = 0.$$

Remark 8.123. *The generalization of the estimate of the interpolation operator is known as the Bramble-Hilbert lemma from 1970 (see e.g. [24]).*

Proof. The proofs use similar techniques as before. However, to get started, let $u \in H^1$. In 1D we discussed that H^1 functions are continuous. We therefore have:

$$\|i_h u\|_{L^2} \leq \max_x |i_h u(x)| \leq \max_x |u(x)| \leq C\|u\|_{H^1}.$$

Furthermore, we have

$$|i_h u|^2_{H^1} = \int_{x_j}^{x_{j+1}} |(i_h u)'(x)|^2\,dx = \frac{\Big(u(x_{j+1}) - u(x_j)\Big)^2}{h} = \frac{1}{h}\Big(\int_{x_j}^{x_{j+1}} u'(x)\,dx\Big)^2 \leq \int_{x_j}^{x_{j+1}} |u'(x)|^2\,dx = |u|^2_{H^1}.$$

As in the other Lemma 8.121, we sum over all elements and obtain

$$\|i_h u\|_{H^1} \leq C|u|_{H^1} \leq C\|u\|_{H^1}.$$

To obtain the second result, we use again (see the other Lemma 8.121):

$$\begin{aligned} u(x) - i_h u(x) &= u(x) - \Big(u(x_j) + \frac{u(x_{j+1}) - u(x_j)}{x_{j+1} - x_j}(x - x_j)\Big) \\ &= \int_{x_j}^{x} u'(t)\,dt - \frac{x - x_j}{x_{j+1} - x_j}\int_{x_{j+1}}^{x_j} u'(t)\,dt \\ &\leq 2\int_{x_j}^{x_{j+1}} |u'(x)|^2\,dx. \end{aligned}$$

We then take again the square, integrate, use Cauchy-Schwarz and finally sum over all elements to obtain

$$\|u - i_h u\|_{L^2} \leq Ch|u|_{H^1}.$$

To establish the third result, let $\varepsilon > 0$. Since C_c^∞ is dense in H^1, we have for all $u \in H^1$:

$$\|u' - v'\|_{L^2} \leq \varepsilon, \quad \text{for } v \in C_c^\infty.$$

With our first estimate on the interpolation error we have:

$$\|i_h u' - i_h v'\|_{L^2} \leq C\|u' - v'\|_{L^2} \leq C\varepsilon.$$

For sufficiently small h we also have (Lemma 8.121, 2nd statement)

$$|v - i_h v|_{H^1} \leq \varepsilon.$$

These results can be used in our final estimate:

$$\|u' - (i_h u)'\|_{L^2} \leq \|u' - v'\|_{L^2} + \|v' - (i_h v)'\|_{L^2} + \|(i_h u)' - (i_h v)'\|_{L^2} \leq 3C\varepsilon,$$

which yields the desired result. □

With the help of the previous three lemmas, we obtain the main result:

Theorem 8.124. *Let $u \in H_0^1$ and $u_h \in V_h$ be the solutions of the continuous and discrete Poisson problems. Then, the finite element method using linear shape functions converges:*

$$\lim_{h\to 0} \|u - u_h\|_{H^1} = 0.$$

Moreover, if $u \in H^2$ (for instance when $f \in L^2$ and in higher dimensions when the domain is sufficiently smooth or polygonal and convex), we have

$$\|u - u_h\|_{H^1} \leq Ch\|u''\|_{L^2} = Ch\|f\|_{L^2}.$$

Thus the convergence in the H^1 norm (the energy norm) is linear and depends continuously on the problem data.

Proof. The first part is proven by using Lemma 8.119 applied to Lemma 8.116, which yields the first part of the assertion. The estimate is based on the Céa lemma:

$$\|u - u_h\|_{H^1} \leq C\|u - \phi_h\| \leq C\|u - i_h u\|_{H^1} \leq Ch|u|_{H^2} = O(h).$$

In the last estimate, we used again Lemma 8.119. □

Corollary 8.125. *We have*

$$\|u - u_h\|_{L^2} \leq Ch\|u''\|_{L^2} = Ch\|f\|_{L^2} = O(h).$$

Here, weakening the norm to L^2 does not seem to have any positive effect on the convergence order, despite that we managed to proof $\|u - i_h u\|_{L^2} = O(h^2)$ in Lemma 8.121, 1st statement.

Proof. Follows immediately from

$$\|u - u_h\|_{H^1} \leq Ch\|u''\|_{L^2} = Ch\|f\|_{L^2}.$$

Now we use, Poincaré:

$$\|u - u_h\|_{L^2} \leq C\|u - u_h\|_{H^1}$$

Thus:

$$\|u - u_h\|_{L^2} \leq C\|u - u_h\|_{H^1} \leq Ch\|u''\|_{L^2} = Ch\|f\|_{L^2} = O(h).$$

Why does

$$\|u - u_h\|_{L^2} \leq C\|u - i_h u\|_{L^2} \leq Ch^2\|u''\|_{L^2}$$

not work? The reason is that Céa's lemma only holds for the natural norm V of the problem setting, here $V = H^1$. Thus, we cannot justify that

$$\|u - u_h\|_{L^2} \leq C\|u - i_h u\|_{L^2}$$

holds true. For this reason our final result is

$$\|u - u_h\|_{L^2} = O(h).$$

□

8.13.3 An improved L^2 estimate - the Aubin-Nitsche trick

Carrying out numerical simulations, see for instance Section 8.16.3.2, we observe $O(h^2)$ in the L^2 norm. This seems one order better than proofed in Corollary 8.125. Actually, according to the L^2 interpolation estimate, we would have expected $O(h^2)$, but could not proof it. In this section, we establish that not only the interpolation estimate is $O(h^2)$, but indeed the approximation error $\|u - u_h\|$ as well.

Proposition 8.126. *Let $u \in H_0^1$ and $u_h \in V_h$ be the solutions of the continuous and discrete Poisson problems. Then:*

$$\|u - u_h\|_{L^2} \leq Ch^2 \|u''\|_{L^2} = Ch^2 \|f\|_{L^2} = O(h^2).$$

Proof. The goal is to obtain one order better than in the previous subsection. To this end, we define an auxiliary problem (the Aubin-Nitsche trick) that considers the error between the exact solution u and the discrete solution u_h: Find $w \in V$ such that

$$a(w, \phi) = \int_\Omega (u - u_h)\phi \, dx \quad \text{for all } \phi \in V.$$

We set $\phi = u - u_h \in V$, indeed since $u \in V$ and $V_h \subset V$ the variable ϕ is in V, and obtain

$$a(w, u - u_h) = \int_\Omega (u - u_h)^2 \, dx = \|u - u_h\|_{L^2}^2.$$

On the other hand we still have Galerkin orthogonality:

$$a(u - u_h, \phi_h) = 0 \quad \forall \phi_h \in V_h.$$

Herein, we take the interpolant $\phi_h := i_h w \in V_h$ and obtain

$$a(u - u_h, i_h w) = 0 \quad \forall \phi_h \in V_h.$$

Combining these two findings, we get:

$$a(w, u - u_h) = a(w - i_h w, u - u_h) = \|u - u_h\|_{L^2}^2.$$

To get a better estimate we now proceed with the continuity of the bilinear form:

$$\|u - u_h\|_{L^2}^2 = a(w - i_h w, u - u_h) \leq \gamma \|w - i_h w\|_{H^1} \|u - u_h\|_{H^1}. \tag{124}$$

The previous interpolation estimate in Theorem 8.124 can now be applied to the auxiliary variable $w \in V$:

$$\|w - i_h w\|_{H^1} \leq Ch \|w''\|_{L^2} = Ch \|u - u_h\|_{L^2}. \tag{125}$$

The last estimate holds true thanks to the auxiliary problem, which reads in strong form (key word: fundamental lemma of calculus of variations!):

$$-w'' = u - u_h \quad \text{in } \Omega, \quad w = 0 \text{ on } \partial\Omega.$$

and in which we need again that Ω is sufficiently regular, which still holds from the original problem of course, and also that the right hand side is in L^2. But the latter one is also okay since $u \in V$ (in particular in L^2 and $u_h \in V_h$ with $V_h \subset V$.

Plugging (125) into (124) yields on the one hand:

$$\|u - u_h\|_{L^2}^2 \leq \gamma Ch \|u - u_h\|_{L^2} \|u - u_h\|_{H^1}$$

therefore

$$\|u - u_h\|_{L^2} \leq \gamma Ch \|u - u_h\|_{H^1}.$$

On the other hand, we know from Theorem 8.124 that

$$\|u - u_h\|_{H^1} \leq Ch \|f\|_{L^2}.$$

Plugging the last equation into the previous equation brings us to:

$$\|u - u_h\|_{L^2} \leq \gamma Ch \|u - u_h\|_{H^1} \leq C^2 h^2 \|f\|_{L^2} = O(h^2).$$

□

8.13.4 Analogy between finite differences and finite elements

Having established the numerical analysis, we can draw a short comparison to finite differences:

- FD: convergence = consistency + stability
- FE: convergence = interpolation + coercivity

8.14 Numerical analysis in 2D and higher dimensions

The 2D and higher dimension results are much more involved and for the detailed proofs, we refer to the literature [24, 35, 106].

Definition 8.127 ([24], Chapter 5)**.** *Let Ω be a polygonal domain in $\mathbb{R}^2$ such that we can approximate this domain with quadrilaterals or triangles. A triangulation $\mathcal{T}_h$ with M elements $K_i, i = 1, \dots, M$ is **admissible** when*

1. *$\bar{\Omega} = \bigcup_{i=1}^{M} K_i$.*
2. *If $K_i \cap K_j$ results in one single point, it is a corner point of K_i and K_j.*
3. *If $K_i \cap K_j$ results in more than point, then $K_i \cap K_j$ is an edge (or face) of K_i and K_j.*

Definition 8.128 (Quasi-uniform, shape regular)**.** *A family of triangulations $\mathcal{T}_h$ is called quasi-uniform or shape regular, when there exists a $\kappa > 0$ such that each $K_i \in \mathcal{T}_h$ contains a circle with radius ρ_K with*

$$\rho_K \geq \frac{h_K}{\kappa}$$

where h_K is half diameter of the element K. The diameter of K is defined as the longest side of K.

Definition 8.129 (Uniform triangulation)**.** *A family of triangulations $\mathcal{T}_h$ is called uniform, when there exists a $\kappa > 0$ such that each $K_i \in \mathcal{T}_h$ contains a circle with radius ρ_K with*

$$\rho_K \geq \frac{h}{\kappa}$$

where $h = \max_{K \in \mathcal{T}_h} h_K$.

Remark 8.130. *Geometrically speaking, the above conditions characterize a **minimum angle condition** , which tells us that the largest angle must not approach $180°$ in order to avoid that triangles become too thin. The constant β is a measure for the smallest angle in K.*

Remark 8.131. *For adaptive mesh refinement (see Section 9.19.4) in higher dimensions, i.e., $n \geq 2$, one has to take care that the minimum angle condition is fulfilled when refining the mesh locally.*

Definition 8.132. *Let $\mathcal{T}_h$ a decomposition of Ω and $m \geq 1$. Then, we define the norm*

$$\|u\|_m := \sqrt{\sum_{K_j \in \mathcal{T}_h} \|u\|^2_{m,K_j}}.$$

Remark 8.133. *We recall that for $m \geq 2$, the Sobolev space H^m is compactly embedded into $C^0(\Omega)$. Consequently in these cases, we can construct a unique interpolation operator $i_h u$. This allows us to define $i_h u$ and to estimate the interpolation error $\|u - i_h u\|$ by higher-order norms.*

8.14.1 The Bramble-Hilbert-Lemma and an interpolation estimate

Lemma 8.134 (Bramble-Hilbert-Lemma)**.** *Let us work in $\mathbb{R}^2$ in order to have the standard Lagrange interpolation. Higher dimensions with other interpolation operators yield, however, the same results. Let now $\Omega \subset \mathbb{R}^2$ a domain with Lipschitz boundary. Let $r \geq 2$ and L a bounded, linear mapping from H^r to a normed space Y. If $P_{r-1} \subset ker(L)$, then it holds for $c > 0$ (i.e., $c(\Omega) \cdot \|L\| \geq 0$):*

$$\|Lu\| \leq c|u|_{H^r} \quad \forall u \in H^r(\Omega).$$

Proof. Braess [24] p. 74. □

Proposition 8.135 ([24], Chapter 6)**.** *Let $r \geq 2$ and $\mathcal{T}_h$ a quasi-uniform (i.e., shape-regular) triangulation of Ω. Then the interpolation via piecewise polynomials of degree $r-1$ yields*

$$\|u - i_h u\|_{H^m} \leq ch^{r-m}|u|_{H^r}, \quad u \in H^r, \quad 0 \leq m \leq r.$$

Proof. See [24][Chapter 6]. Be careful, these are long and nontrivial proofs in which one learns however a lot of things. □

8.14.2 Inverse estimates

Working in the continuous function spaces allows for the previous approximation results. Specifically, we have

$$\|u - i_h u\|_{H^m} \leq ch^{r-m}|u|_{H^r},$$

where $m \leq r$. In other words: the approximation error is measured in a coarser norm than the function itself. On the discrete level, we can show results the other way around: the finer norm of finite elements can be estimated by coarser norms.

Proposition 8.136 (Inverse estimate)**.** *Let $V_h := V_h^{(k)}$ be discrete function spaces based on uniform decompositions. Then, there exists a constant $c := c(\kappa, k, r)$ such that for $0 \leq m \leq r$:*

$$\|u_h\|_r \leq c\, h^{m-r}\|u\|_m \quad \forall u_h \in V_h.$$

8.14.3 Error estimates

Definition 8.137. *Let $m \geq 1$ and $H_0^m \subset V \subset H^m$. Let $a(\cdot,\cdot)$ be a V-elliptic bilinear form. The variational problem, find $u \in V$ such that*

$$a(u, \phi) = l(\phi) \quad \forall \phi \in V$$

is H^s regular if for each $f \in H^{s-2m}$ a solution $u \in H^s(\Omega)$ exists. Furthermore, it holds

$$\|u\|_{H^s} \leq c\|f\|_{H^{s-2m}}$$

Here, we assume $s \geq 2m$. For negative norms, this last condition can be omitted.

Theorem 8.138. *Let $\mathcal{T}_h$ be a quasi-uniform decomposition of Ω. Then it holds for $u_h \in V_h^{(k)}, k = 1, 2, 3$ and triangular or quadrilaterals elements:*

$$\|u - u_h\|_{H^1} \leq ch\|u\|_{H^2} \leq ch\|f\|_{L^2} = O(h)$$

Using a duality argument (Aubin-Nitsche; see our 1D proofs), we obtain:

Theorem 8.139. *Let the previous assumptions hold true, then:*

$$\|u - u_h\|_{L^2} \leq ch^2\|f\|_{L^2} = O(h^2).$$

8.15 Numerical analysis: influence of numerical quadrature, first Strang lemma

Previously, we studied the Céa-Lemma in terms of approximation theory. Now, a second error estimate for $\tilde{u}_h$ becomes important which is a consequence of **numerical quadrature**, we denote it **error of consistence**, named **first Lemma from Strang**.

Our focus is combining both errors and give some answers about the order of convergence in the H_1-norm $||\cdot||_1$, that means

$$||u - \tilde{u}_h||_1 \leq \underbrace{||u - u_h||_1}_{\text{Céa}} + \underbrace{||u_h - \tilde{u}_h||_1}_{\text{Strang}}$$

This investigation leads to practical results in solving integrals with numerical integration. One could say which degree of quadrature formula is necessary integrating finite elements at a given degree.

This discussion covers two possible ways: First, we study interpolation by Lagrange with exact integration. The second way illustrates approximation theory for numerical quadrature formulae, the most common way solving integrals in FEM.

Therefore, we study an elliptic variational problem in two dimensions. This includes the model problem (Poisson problem). The results also hold for elliptic PDE's of second order in 1D.

Formulation 8.140. *Let $\Omega \subset \mathbb{R}^2$ be a bounded, convex, polygonal domain. The task is finding a weak solution $u \in H_0^1(\Omega)$ for an elliptic boundary value problem in $\mathbb{R}^2$ with homogeneous Dirichlet conditions*

$$u \in H_0^1(\Omega): \quad a(u,v) = (f,v)_0 \quad \text{for } v \in H_0^1(\Omega)$$

with

$$a(u,v) = \sum_{i,k=1}^{2} \int_\Omega a_{ik} \partial_k u \partial_i v \, dx \tag{126}$$

and coefficient functions $a_{ik} := a_{ik}(x)$.

We assume a regular triangulation $\mathbb{T}_h = \{T\}$ from $\overline{\Omega}$ with properties

i) $\overline{\Omega} = \bigcup_i T_i$

ii) $T_i^\circ \cap T_\nu^\circ = \varnothing\,, \quad \nu \neq i$

For more information see [24] p. 58.

8.15.1 Approximate solution of A and b

There is often no chance or an enormous computational cost solving matrix A and right-hand side b. In principle we have the reasons

i) The first case puts $a_{ij} = \delta_{ij}$ in (126), so the computation uses integration formulae for polynomials of degree $2m-2$. Exact formulae for higher m need many computations but convergence is sure for quadrature rules of lower order.

ii) In general, the coefficient functions a_{ij} are not constant. Therefore, exact computations for the elements of A_{ij} and b_i are not possible.

iii) Numerical integration is the favorite way to determine isoparametric elements.

The next ideas where introduced by Strang and generalize the Céa-Lemma. Already known is the error of approximation and now a second estimate becomes important, named *error of consistence*.

Calculations will be done on a reference element subsequently affine-transformed on each cell (triangle).

8.15.1.1 Idea, deriving an approximate solution The matrix $A = (A_{ij})_{i,j=1}^N$ and right-hand-side $b = (b_i), i = 1, \ldots, N$ leads to the linear equation system

$$Au = b$$

Instead of solving this one, an approximate linear equation system is given by

$$\tilde{A}\tilde{u} = \tilde{b} \tag{127}$$

with $\tilde{A} = (\tilde{A}_{ij})_{i,j=1}^N$ and $\tilde{b} = (\tilde{b}_i), i = 1, \ldots, N$. The solution of (127) brings us the approximate solution vector

$$\tilde{u}_h = \sum_{i=1}^N \tilde{\alpha}_i w_i \in S_h^m .$$

Now the error

$$e_h = u_h - \tilde{u}_h$$

arises. Next we give an idea for the corresponding variational formulation of (127). We work with two linear combinations of S_h^m:

$$v_h = \sum_{i=1}^N \xi_i w_i \quad \text{and} \quad w_h = \sum_{j=1}^N \eta_j w_j$$

These are necessary for the approximate bilinear form $\tilde{a}(v_h, w_h)$ and the approximate right-hand-side $\tilde{l}(v_h)$:

$$\tilde{a}(v_h, w_h) = \sum_{i,j=1}^N \tilde{A}_{ij}\xi_i\eta_j \quad \text{and} \quad \tilde{l}(v_h) = \sum_{i=1}^N \tilde{b}_i\xi_i \tag{128}$$

This leads to the variational formulation $(\tilde{V})$

$$\tilde{u}_h \in S_h^m : \quad \tilde{a}(\tilde{u}_h, v_h) = \tilde{l}(v_h) \quad \forall v_h \in S_h^m \qquad \Leftrightarrow \qquad \tilde{A}\tilde{u} = \tilde{b}$$

The following lemma describes an error estimate for $u_h - \tilde{u}_h$.

Lemma 8.141. *(first Lemma from Strang)*
The approximate bilinear form $\tilde{a}$ and the linear functional $\tilde{l}$ satisfying the condition

$$||v_h||_1^2 \leq \gamma\tilde{a}(v_h) \qquad \textit{(Uniform ellipticity of } \tilde{a}) \tag{129}$$

uniformly in $0 < h \leq h_0$ and also holds the estimate

$$|(a - \tilde{a})(u_h, v_h)| + |(f, v_h)_0 - \tilde{l}(v_h)| \leq ch^\tau \cdot ||v_h||_1 , \quad v_h \in S_h^m \tag{130}$$

Then the problems $(\tilde{V})$ have unique solutions $\tilde{u} \in S_h^m$. The quantitative error estimate of the Ritzapproximation $u_h \in S_h^m$ is given by

$$||u_h - \tilde{u}_h||_1 \leq c\gamma h^\tau$$

Proof.
The unique solvability may be proofed with the Theorem of Lax-Milgram. See [24] p. 37 respectively [108] p.41. We derive the quantitative error estimation by the following calculation. Denote $e_h = v_h = u_h - \tilde{u}_h$,

$$\begin{aligned}
\tilde{a}(e_h) &= \tilde{a}(e_h, e_h) = \tilde{a}(u_h - \tilde{u}_h, e_h) = \tilde{a}(u_h, e_h) - \tilde{a}(\tilde{u}_h, e_h) \\
&= \tilde{a}(u_h, e_h) - a(u_h, e_h) + a(u_h, e_h) - \tilde{a}(\tilde{u}_h, e_h) \\
&= (\tilde{a} - a)(u_h, e_h) + a(u_h, e_h) - \tilde{a}(\tilde{u}_h, e_h) \\
&= (\tilde{a} - a)(u_h, e_h) + (f, e_h)_0 - \tilde{l}(e_h) \\
&\leq ch^\tau ||e_h||_1
\end{aligned}$$

The last estimation follows from second assumption (130). Since we have (129), we can conclude:

$$||e_h||_1^2 \leq \gamma\tilde{a}(e_h) \leq \gamma ch^\tau ||e_h||_1$$

Division by $||e_h||_1$ shows the assertion.

□

With the aid of (128) we have a corresponding formulation for uniform ellipticity of $\tilde{a}$:

$$\tilde{a}(v_h) = \sum_{i,j} \tilde{A}_{ij}\xi_i\xi_j \geq \gamma_1|\xi|^2 \quad \forall \xi \in \mathbb{R}^n,\, \gamma_1 > 0 \tag{131}$$

Hence the coefficient matrix is uniformly definite in the variable x.

We point out two different ways for an approximate determination of A and b. In this context, we assume that the coefficient functions a_{ij} and the power vector f are sufficiently smooth,

$$a_{ij}, f \in L^\infty(E), \quad i,j = 1,2$$

Secondly, we work with an *affine family* of finite elements, so we have the polynomial space $P_m(E)$ for the reference element. Interpolation and quadrature formulae were derived on that reference element and later transformed on each cell of Ω.

8.15.2 Interpolation with exact polynomial integration

There is a triangulation $\mathbb{T}_h$. Each triangle $T \in \mathbb{T}_h$ satisfies polynomial interpolation for a_{ij} and f with

$$\tilde{a}_{ij}, \tilde{f} \in P_{r-1}(T), \quad i,j = 1,2$$

The error is determined by the remainder and we find the uniform estimation

$$||a_{ij} - \tilde{a}_{ij}||_\infty + ||f - \tilde{f}||_\infty = \mathcal{O}(h^r) \tag{132}$$

This error is independent from step width h and holds for functions $a_{ij}, f \in C^r(T)$. Higher differentiability leads not to a better error estimation. See [69] p. 194ff.

We compute the elements of $\tilde{A}$ and right-hand-side $\tilde{b}$ with

$$\tilde{A}_{ij} = \int_\Omega \sum_{\nu,\mu=1}^{2} \tilde{a}_{\nu\mu}\partial_\mu w_i\, \partial_\mu w_j\, dx, \quad i,j = 1,\dots,N\,,$$

$$\text{and} \quad \tilde{b}_i = \int_\Omega \tilde{f} w_i\, dx, \quad i = 1,\dots,N\,.$$

The advantage of these elements is exact integration. Therefore we have to evaluate the following integrals,

$$\int_T x^\beta\, dx, \quad 0 \leq |\beta| \leq 2m + r - 3$$

Proof. Sketch. We give an answer that the degree of the polynomials must be $0 \leq |\beta| \leq 2m + r - 3$. The assumption says $w_i \in S_h^m$ which means $w_i|_T \in P_m(T)$. The coefficient function $\tilde{a}_{ij}$ is a polynomial of $\in P_{r-1}(T)$. Then

$$\tilde{a}_{\nu,\mu}\partial_\nu w_i\, \partial_\mu w_j \in P_{(r-1)+2m-2}(T)$$

The proof is finished. □

An error estimation is given by Lemma (8.141). First task is to check the assumptions. Consider $v_h, w_h \in S_h^m$:

$$
\begin{aligned}
|(a-\tilde{a})(v_h,w_h)| &= |a(v_h,w_h)-\tilde{a}(v_h,w_h)| && (133)\\
&= \left| \int_G \sum_{\nu,\mu=1}^{2} a_{\nu\mu}\partial_\mu v_h\, \partial_\nu w_h\, dx - \int_G \sum_{\nu,\mu=1}^{2} \tilde{a}_{\nu\mu}\partial_\mu v_h\, \partial_\nu w_h\, dx \right| && (134)\\
&= \left| \int_G \sum_{\nu,\mu=1}^{2} (a_{\nu\mu}-\tilde{a}_{\nu\mu})\partial_\mu v_h\, \partial_\nu w_h\, dx \right| && (135)\\
&\leq c||a-\tilde{a}||_\infty \cdot \left| \int_G \sum_{\nu,\mu=1}^{2} \partial_\mu v_h\, \partial_\nu w_h\, dx \right| && (136)\\
&\overset{\text{C.S.}}{\leq} c||a-\tilde{a}||_\infty \cdot ||v_h||_1 \cdot ||w_h||_1 && (137)\\
&\overset{(132)}{\leq} ch^r||v_h||_1 \cdot ||w_h||_1 && (138)
\end{aligned}
$$

We proof the second part

$$
\begin{aligned}
|(f,v_h)-\tilde{l}(v_h)| &= \left| \int_G f\, v_h\, dx - \int_G \tilde{f}\, v_h\, dx \right| && (139)\\
&= \left| \int_G (f-\tilde{f})\, v_h\, dx \right| && (140)\\
&\leq c||f-\tilde{f}||_\infty \cdot ||v_h||_\infty && (141)\\
&\overset{(132)}{\leq} ch^r \cdot ||v_h||_1 && (142)
\end{aligned}
$$

We complement the proof showing (129),

$$
\begin{aligned}
||v_h||_1^2 &\leq c\, a(v_h) && \text{(V-ellipticity)} && (143)\\
&\leq c|a(v_h)-\tilde{a}(v_h)+\tilde{a}(v_h)| && \text{(expansion with } \tilde{a}(v_h)\text{)} && (144)\\
&\leq c|a(v_h)-\tilde{a}(v_h)|+c|\tilde{a}(v_h)| && \text{(triangle inequality)} && (145)\\
&= c|(a-\tilde{a})(v_h)|+c|\tilde{a}(v_h)| && && (146)\\
&\leq ch^r||v_h||_1^2 + c\,\tilde{a}(v_h) && \text{(set } v_h = w_h \text{ in (138))} && (147)
\end{aligned}
$$

The last estimate is a result from $\tilde{a}(v_h) > 0$. For sufficient small step size $0 < h \leq h_0$ we can conclude

$$||v_h||_1^2 \leq c\,\tilde{a}(v_h)$$

Now, Lemma (8.141) delivers the error of the approximate Ritzapproximation

$$||u_h - \tilde{u}_h||_1 \leq ch^r$$

8.15.2.1 Total error We recall the roles of the different solutions

- u : original function which is to approximate,
- the function u is approached by the solution u_h of the Ritz-procedure,
- for lower computational cost, u_h is approximated itself by $\tilde{u}_h$.

The triangle inequality shows

$$||u-\tilde{u}_h||_1 \leq ||u-u_h||_1 + ||u_h-\tilde{u}_h||_1 = \mathcal{O}(h^m + h^r)$$

The order for optimal convergence is $r \geq m$. The error of the interpolating functions can be neglected in comparison with the procedure-error if $r > m$.

8.15.3 Integration with numerical quadrature

In this context numerical integration formulae based on interpolated functions. The most common formulae is

$$f_T(g) = \int_T g\,dx \sim Q_T(g) = \sum_{i=1}^{L} \omega_i g(\xi_i) \tag{148}$$

with distinct quadrature points $\xi_i \in T$ and quadrature weights ω_i. For construction of these formulae, please visit other literature.

8.15.3.1 Affine transformation rules We define the unit triangle E as the reference element. Let $\sigma_T : E \to T$ be affine-linear mapping of E onto the triangle T. Then we have the properties

$$x = \sigma_T(\hat{x}) = B_T\hat{x} + b_T, \quad x \in T,\, \hat{x} \in E$$

with the functional matrix B_T.

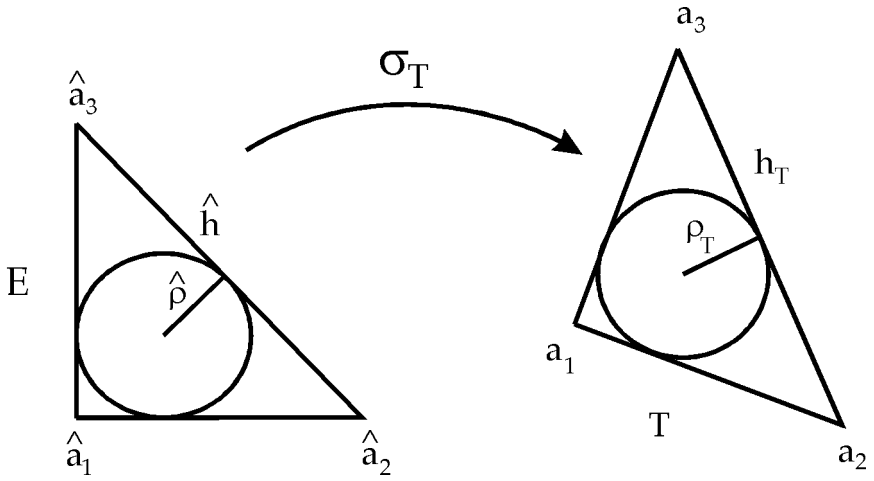


Figure 27: Reference element E and arbitrary cell (triagle) T

Example 8.142. *Let $E = conv\{\hat{a}_1, \hat{a}_2, \hat{a}_3\}$ with $\hat{a}_1 = (0,0)^T, \hat{a}_2 = (1,0)^T, \hat{a}_3 = (0,1)^T$. The triangle T has the coordinates $a_1 = (3, 1/4), a_2 = (4, -1/2), a_3 = (17/4, 1)$.*

A bijective mapping between E and a triangle T is given by $\sigma_T : E \to T$ with $\sigma_T(\hat{x}) = B_T\hat{x} + b$, where $b = a_1$. The functional matrix is defined by

$$B_T = (a_2 - a_1, a_3 - a_1) = \begin{pmatrix} 1 & \frac{5}{4} \\ -\frac{3}{4} & \frac{3}{4} \end{pmatrix}$$

The determinant of B_T is of order h^2. That means $detB_T = \mathcal{O}(h^2)$.

We obtain the relation, also known as "affine-equivalence" between two Lagrange elements,

$$\hat{v}(\hat{x}) := v(x), \quad \hat{x} = \sigma_T^{-1}x,\, x \in T$$

between functions on E and T. The polynomial space for the reference element is charakterized by $P_m(E)$. A (linear) quadrature rule on E satisfies

$$Q_E(g) = \sum_{i=1}^{L} \hat{\omega}_i g(\hat{\xi}_i), \quad \hat{\omega}_i := \int_E \hat{L}_i(\hat{x})d\hat{x},\, \hat{L}_i(\hat{x}) = \prod_{k=0}^{n} \frac{\hat{x} - x_k}{x_i - x_k}$$

with quadrature points $\hat{\xi}_i \in E$ and positive weights $\hat{\omega}_i$, $i = 1, \ldots, L$.

Affine transformation of QF on E gives quadrature rules on each element T with

$$Q_T(g) = \sum_{i=1}^{L} \omega_i g(\xi_i)$$

with quadrature points $\xi_i = \sigma_T \hat{\xi}_i$ and weights $\omega_i = |\mathrm{det} B_T| \hat{\omega}_i, i = 1, \ldots, L$. Then

$$Q_T(g) := \sum_{i=1}^{L} \omega_i g(\xi_i) = \sum_{i=1}^{L} |\mathrm{det} B_T| \hat{\omega}_i \hat{g}(\hat{\xi}_i) = |\mathrm{det} B_T| \, |Q_E(\hat{g})|$$

There is a short introduction of the very important transformation rule for integrals. It follows from the substitution rule in 1D.

Lemma 8.143. *Let T_1 and T_2 open subsets of $\mathbb{R}^n$ and $\sigma : T_1 \to T_2$. Then, and only then, the function f on T_2 is integrable if $(f \circ \sigma) \cdot |det\sigma'|$ is integrable over T_1. It is*

$$\int_{T_1} f(\sigma(x)) \cdot |det\sigma'| \, dx = \int_{T_2} f(y) \, dy$$

Proof. See [86] p. 299. It follows directly for an affine mapping $\sigma\hat{x} = B\hat{x} + b$, with derivation $D(\sigma\hat{x}) = D(B\hat{x} + b) = B$, that

$$\int_T v(x) \, dx = |\mathrm{det} B_T| \int_E \hat{v}(\hat{x}) \, d\hat{x}$$

□

The next two estimations can directly derived from the geometry of our triangulation. Remember the use of triangles in a regular triangulation. "Regular" means, that

$$\varrho_T \geq c h_T \quad \forall h, T, \, c > 0$$

where h_T charakterises the maximum diameter for any triangle, that means the longest side. In general we have the following two situations

$$||B_T||_2 \leq \frac{h_T}{\hat{\varrho}} = c_1 h_T \quad \text{and} \quad ||B_T^{-1}||_2 \leq \frac{\hat{h}}{\varrho_T} = c_0 \varrho_T^{-1} \tag{149}$$

The symbol $||B||_2$ means

$$||B||_2 = \left(\sum_{i,k} |b_{ik}|^2 \right)^{\frac{1}{2}}$$

Furthermore there is an estimation for determinants as follows

$$||B_T^{-1}||_2^{-n} \leq |\mathrm{det} B_T| \leq ||B_T||_2^n$$

which may be proofed with the Laplace expansion theorem. Then

$$c_0 \varrho_T^n \leq |\mathrm{det} B_T| \leq c_1 h_T^n$$

and

$$||B_T||_2 \, ||B_T^{-1}||_2 \leq C \frac{h}{\varrho}$$

It follows in 2D ($n = 2$)

$$C_0 h_T^2 \leq \frac{1}{||B_T^{-1}||_2^2} \leq |\mathrm{det}(B_T)| \leq ||B_T||_2^2 \leq C_1 h_T^2 \tag{150}$$

At last we mention

$$c_0 |\widehat{\nabla} \hat{v}(\hat{x})| \leq h_T |\nabla v(x)| \leq c_1 |\widehat{\nabla} \hat{v}(\hat{x})|, \quad x \in T, \hat{x} = \sigma_T^{-1} x \tag{151}$$

Proof. Sketch. Estimation (151) follows from the compound rule of three

$$\widehat{\nabla}\hat{v}(\hat{x}) = B_T^T \nabla v(x) \tag{152}$$

We have

$$|\nabla v(x)| = |\nabla v(\sigma(\hat{x}))| = |B^{-T}\widehat{\nabla}\hat{v}(\hat{x})| \leq ||B^{-T}||_2 |\widehat{\nabla}\hat{v}(\hat{x})|$$

and on the other hand

$$|\widehat{\nabla}\hat{v}(\hat{x})| = |B^T \nabla v(x)| \leq ||B^T||_2 |\nabla v(x)|$$

Working with (149) show (151). □

8.15.3.2 Numerical quadrature We set QF as an abbreviation for *quadrature formula*.

Definition 8.144. *(Order of a QF)*
A QF $Q_T(\cdot)$ on T is said to be of order r if it integrates exactly all polynomials of degree $r-1$,

$$Q_T(p) = \int_T p\,dx, \quad p \in P_{r-1}(T)$$

Another equivalent formulation is given by

$$Q_T(g) = \int_T P_T g\,dx, \quad P_T g \in P_{r-1}(T)$$

where P_T is an interpolating operator for g. Then, $Q_T(\cdot)$ is named as an interpolatary approximation of $f_T(\cdot)$.
The first result gives an estimation for the error of numerical quadrature. Therefore we need the seminorm

$$|u|_{r,1,T} = \sum_{|\alpha|=r} \int_T |\partial^\alpha u|\ dx$$

and we mention that $C^r(T)$ is dense in $H^{r,1}(T)$.

Lemma 8.145. *(Quantitative error of numerical quadrature)*
Assume $Q_T(g)$ is a QF of order $r \geq 3$ of the function $g \in H^{r,1}(T)$. Then

$$|f_T(g) - Q_T(g)| \leq ch^r |g|_{r,1,T} \quad \forall T \tag{153}$$

for constant c which is independent of T and h

Proof. The requirement $r \geq 3$ will be proofed with the inequality from Sobolev which is a consequence of the embedding theorem. Note, that the case $r = 2$ is also possible but difficult to prove. The linear functional f_T is continuous with respect to the L^1-norm,

$$|f_T(g)| = \left| \int_T g\,dx \right| \leq \int_T |g|\,dx = |g|_{0,1,T}\,, \quad g \in L^1(T) \tag{154}$$

It follows for $g \in H^{r,1}(T)$ and all T

$$\begin{aligned} |f_T(g) - Q_T(g)| &= |f_T(g) - f_T(P_T g)| \qquad \text{(interpolated approximation of } f_T\text{)} \\ &= |f_T(g - P_T g)| \\ &\overset{(154)}{\leq} |g - P_T g|_{0,1} \\ &\leq ch^r |g|_{r,1} \qquad \text{(interpolation estimation)} \end{aligned}$$

We finished proof. □

Definition 8.146. *(Admissibility of a QF)*
Each polynomial $q \in P_{m-1}(E)$, integrated by Q_E, has the property

$$For \quad q(\hat{\xi}_i) = 0,\ i = 1, \dots, L \quad follows \quad q \equiv 0 \tag{155}$$

This property ensures that the Lagrange interpolation has a unique solution. The necessary condition for (155) requires that the number of interpolation points L is greater or equal to $\dim P_{m-1}$.

We need further results like a new norm, and after that definition, we give an estimation between QF and integral.

Definition 8.147. *(Norm on $P_{m-1}(E)/P_0(E)$)*
On the finite dimensional quotient space $P_{m-1}(E)/P_0(E)$ is a norm defined by

$$|||\hat{q}||| := \left(\sum_{i=1}^{L} \hat{\omega}_i \sum_{j=1}^{2} (\partial_j q)^2(\hat{\xi}_i) \right)^{\frac{1}{2}}$$

Proof. We have to proof all attributes of a norm. But the friendly reader wants to show three properties. We only check,

Definiteness.

Property (69) holds. Since the quadrature weights $\hat{\omega}_i$ are positive it follows for $q \in P_{m-1}(E)$,

$$\begin{aligned} &\sum_{i=1}^{L} \hat{\omega}_i \sum_{j=1}^{2} (\partial_j q)^2(\hat{\xi}_i) = 0 \\ \Rightarrow \quad &\partial_j q(\hat{\xi}_i) = 0, \quad 1 \le j \le 2, \quad 1 \le i \le L \end{aligned}$$

Remark that any $\partial_j q$ is an element of $P_{m-2}(E)$ because $q \in P_{m-1}(E)$. Since (69), $\partial_j q$ vanishes at $\hat{\xi}_i$. This implies $\partial_j q \equiv 0$, $j = 1, 2$ which shows the definiteness.

The reader may check

$$||q|| \ge 0, \quad ||\alpha q|| = |\alpha|\,||q||,\ \alpha \in \mathbb{R}, \quad ||p + q|| \le ||p|| + ||q||$$

□

Lemma 8.148. *(Ellipticity of QF)*
For some constant c which is independent of h we have

$$Q_T(|\nabla v|^2) \ge c \int_T |\nabla v|^2 dx \quad \forall v \in S_h^{m-1} \quad \forall T \tag{156}$$

Proof. There is some work to do. First we have the norm of definition (8.147). A second norm on $P_{m-1}(E)/P_0(E)$ is given by $|\hat{q}|_{1,E}$. The finite dimension of $P_{m-1}(E)/P_0(E)$ leads to the equivalence of the two norms. It exists some constant $\hat{C}$, that

$$\hat{C}|\hat{q}|_1^2 \le |||\hat{q}|||, \quad \hat{q} \in P_{m-1}(E)/P_0(E) \tag{157}$$

We compute

$$\begin{aligned} Q_T(|\nabla p|^2) &\overset{(148)}{=} \sum_{i=1}^{L} \omega_i |\nabla p(\xi_i)|^2 \\ &= |\det B_T| \sum_{i=1}^{L} \widehat{\omega}_i |\nabla p(\xi_i)|^2 \quad (\text{with } \omega_i = |\det B_T| \hat{\omega}_i) \\ &\overset{(151)}{\ge} c_0 |\det B_T| \frac{1}{h^2} \sum_{i=1}^{L} \hat{\omega}_i |\widehat{\nabla} \hat{q}(\hat{\xi}_i)|^2 \\ &\ge c_0' \cdot Q_E(|\widehat{\nabla} \hat{q}|^2) \\ &= c_0' |||\hat{q}|||, \quad \hat{q} \in P_{m-1}(E)/P_0(E) \end{aligned}$$

We conclude with the aid of equivalence of the two norms (157)

$$\begin{aligned} Q_T(|\nabla p|^2) &\geq c_0'|||\hat{q}|||^2 \overset{(157)}{\geq} c_0'\hat{C}|\hat{q}|_{1,E}^2 \\ &= c_0|\mathrm{det}B_T|h^{-2}|\hat{q}|_{1,E}^2 = c_0|\mathrm{det}B_T|h^{-2}\int_E |\widehat{\nabla}\hat{q}|^2\,d\hat{x} \\ &= c_0|\mathrm{det}B_T|h^{-2}\int_E |B_T^T\,\nabla q|^2\,d\hat{x} \\ &\geq c_0h^{-2}\int_T ||B_T^T||^2|\nabla q|^2\,dx \\ &\overset{(150)}{\geq} c_0h^{-2}\,C_1h^2\int_T |\nabla q|^2\,dx \\ &= C\int_T |\nabla p|^2\,dx \end{aligned}$$

for $p \in S_h^{m-1}$. Which completes the proof. □

8.15.3.3 Error estimation The central theorem leads to an error estimation of the FEM for numerical quadrature formulae.

We give the definitions of the approximate elements of the matrix $\tilde{A}_{ij}$ and the right-hand side $\tilde{b}_i$

$$\tilde{A}_{ij} = \sum_T Q_T\left(\sum_{\nu,\mu=1}^{2} a_{\mu\nu}\partial_\mu w_i\,\partial_\nu w_j\right), \quad i,j = 1,\ldots,N$$

and

$$\tilde{b}_i = \sum_T Q_T(fw_i), \quad i = 1,\ldots,N$$

The main Theorem is a consequence of the Lemma from Strang (8.141). First, we proof the conditions, then we have the error estimation for $u_h - \tilde{u}_h$.

Theorem 8.149. *Assume the quadrature formula, of order $r \geq 3$,*

$$Q_E(g) = \sum_{i=1}^{L} \hat{\omega}_i g(\hat{\xi}_i)$$

on E, which is admissible for $P_{m-1}(E)$. Let $r \geq m-1$, $m-2$. The degree of the finite elements is $m-1$ that is $u_h, v_h \in S_h^{m-1}$. The coefficient functions $a_{\nu\mu} \in L^\infty(E)$ are sufficiently smooth. Then we have

$$|(a-\tilde{a})(u_h,v_h)| \leq Ch^{r-m+2}||v_h||_1, \quad v_h \in S_h^{m-1}$$

and

$$|(f,v_h)_0 - \tilde{l}(v_h)| \leq Ch^{r-m+2}||v_h||_1, \quad v_h \in S_h^{m-1}$$

and also uniform ellipticity

$$c||v_h||_1^2 \leq \tilde{a}(v_h), \quad v_h \in S_h^{m-1}$$

The constants c, C depending on u.

Proof. Set $m' := m - 1$. Denote $v_h \in S_h^{m'}$:

$$\begin{aligned}
|(a-\tilde{a})(u_h, v_h)| &= |a(u_h, v_h) - \tilde{a}(u_h, v_h)| \\
&= \left| \sum_T \left[\int_T \sum_{\nu,\mu=1}^{2} a_{\nu\mu} \partial_\mu u_h \, \partial_\nu v_h \, dx - Q_T \left(\sum_{\nu,\mu=1}^{2} a_{\nu\mu} \partial_\mu u_h \, \partial_\nu v_h \right) \right] \right| \\
&= \left| \sum_T \sum_{\nu,\mu=1}^{2} \left[\int_T a_{\nu\mu} \partial_\mu u_h \, \partial_\nu v_h \, dx - Q_T \left(a_{\nu\mu} \partial_\mu u_h \, \partial_\nu v_h \right) \right] \right| \\
&\overset{\text{TRI}}{\leq} \sum_T \left| \sum_{\nu,\mu=1}^{2} \left[\int_T a_{\nu\mu} \partial_\mu u_h \, \partial_\nu v_h \, dx - Q_T \left(a_{\nu\mu} \partial_\mu u_h \, \partial_\nu v_h \right) \right] \right| \\
&\overset{(153)}{\leq} ch^r \sum_T \left| \sum_{\nu,\mu=1}^{2} a_{\nu\mu} \partial_\mu u_h \, \partial_\nu v_h \right|_{r,1,T}
\end{aligned}$$

The functions $a_{\nu\mu}$ are bounded with $||a_{\nu\mu}||_\infty \leq c$ which implies

$$|(a-\tilde{a})(u_h, v_h)| \overset{(153)}{\leq} ch^r \sum_T \left| \sum_{\nu,\mu=1}^{2} a_{\nu\mu} \partial_\mu u_h \, \partial_\nu v_h \right|_{r,1,T} \tag{158}$$

$$\overset{\text{C.S.}}{\leq} ch^r \sum_T \left(\sum_{\mu=1}^{2} |\partial_\mu u_h|_{r,1,T}^2 \right)^{\frac{1}{2}} \cdot \left(\sum_{\nu=1}^{2} |\partial_\nu v_h|_{r,1,T}^2 \right)^{\frac{1}{2}} \tag{159}$$

$$\leq ch^r \sum_T ||u_h||_{r+1,T} \cdot ||v_h||_{r+1,T} \tag{160}$$

Now, a function $w_h \in S_h^{m'}$ on each triangle $T \in \mathbb{T}_h$ is a polynomial of $P_{m'}(T)$. Because of differentiability, we compute

$$||w_h||_{r+1,T} = ||w_h||_{m,T} \quad \text{for } r \geq m' \tag{161}$$

The inverse estimation delivers

$$||w_h||_{m',T} \leq ch^{1-m'} ||w_h||_{1,T} \tag{162}$$

Using (161) and (162) for the function v_h show us

$$||v_h||_{r+1,T} \leq ch^{1-m'} ||v_h||_{1,T} \tag{163}$$

We obtain with (161) for u_h

$$||u_h||_{r+1,T} = ||u_h||_{m',T} \tag{164}$$

We put the last two results and the Cauchy-Schwarz inequality in (160),

$$\begin{aligned}
|(a-\tilde{a})(u_h, v_h)| &\leq ch^r \sum_T ||u_h||_{r+1,T} \cdot ||v_h||_{r+1,T} \\
&\overset{\text{C.S.}}{\leq} ch^r \left(\sum_T ||v_h||_{r+1,T}^2 \right)^{\frac{1}{2}} \cdot \left(\sum_T ||u_h||_{r+1,T}^2 \right)^{\frac{1}{2}} \\
&\overset{(163),(164)}{\leq} ch^{r-m'+1} ||v_h||_1 \cdot \left(\sum_T ||u_h||_{m',T}^2 \right)^{\frac{1}{2}} \\
&= ch^{r-m'+1} ||v_h||_1 \cdot ||u_h||_{m'}
\end{aligned}$$

We specify the last term with the Ritzapproximation u_h. Taking an interpolation operator $I_h u \in S_h^{m'}$ for u we conclude

$$\begin{aligned}
||u_h||_{m',T} &= ||u_h - I_h u + I_h u - u + u||_{m',T} \\
&\leq ||u_h - I_h u||_{m',T} + ||I_h u - u||_{m',T} + ||u||_{m',T} \\
&\overset{(162)}{\leq} ch^{1-m'}||u_h - I_h u||_{1,T} + ||I_h u - u||_{m',T} + ||u||_{m',T} \\
&= ch^{1-m'}||u_h - u + u - I_h u||_{1,T} + ||I_h u - u||_{m',T} + ||u||_{m,T} \\
&\leq ch^{1-m'}||u_h - u||_{1,T} + ch^{1-m'}||u - I_h u||_{1,T} + ||I_h u - u||_{m',T} + ||u||_{m',T}
\end{aligned}$$

Remember the result from approximation with I_h:

$$||u - I_h u||_{m'} \leq ch^{t-m'}|u|_t \leq ch^{t-m'}||u||_t \quad \text{for } u \in H^t(\Omega),\ 0 \leq m' \leq t.$$

Then, we have

$$\begin{aligned}
||u_h||_{m',T} &\leq ch^{1-m'}||u_h - u||_{1,T} + c_1||u||_{m',T} + c_2||u||_{m',T} + ||u||_{m',T} \\
&\leq ch^{1-m'}||u_h - u||_{1,T} + C||u||_{m',T}
\end{aligned}$$

The extension on Ω is

$$||u_h||_{m'} = \left(\sum_T ||u_h||^2_{m',T}\right)^{\frac{1}{2}} \leq ch^{1-m'}||u - u_h||_1 + c||u||_{m'}$$

Since $||u - u_h||_1 = \mathcal{O}(h^{m'})$, gives

$$\begin{aligned}
|(a - \tilde{a})(u_h, v_h)| &\leq ch^{r-m'+1}||v_h||_1 \cdot ||u_h||_{m'} \\
&\leq ch^{r-m'+1}||v_h||_1 \left(ch^{1-m'}||u - u_h||_1 + c||u||_{m'}\right) \\
&= ch^{r-2m'+2}||u - u_h||_1||v_h||_1 + ch^{r-m'+1}||v_h||_1 \cdot ||u||_{m'} \\
&= ch^{r-m'+2}||v_h||_1 + c(u)h^{r-m'+1}||v_h||_1 \\
&\leq c(u)h^{r-m'+1}||v_h||_1
\end{aligned}$$

We explain in words: The third line uses the order of convergence $||u - u_h|| = \mathcal{O}(h^{m'})$. The number $||u||_{m'}$ is not important for convergence, so it is some constant factor $c(u)$. The order of convergence of $h^{r-m'+1}$ is not as good as $h^{r-m'+2}$ which completes the last line. Remark that $r \geq 3$ and $r \geq m'$. Short summary of the result ($m' = m - 1$):

$$|(a - \tilde{a})(u_h, v_h)| \leq ch^{r-m+2}||v_h||_1, \quad c = c(u)$$

Analogous computation verifies

$$|(f, v_h)_0 - \tilde{l}(v_h)| \leq ch^{r-m+2}||v_h||_1$$

The last estimation shows uniform ellipticity of $\tilde{a}$. Let $v_h \in S_h^{m'}$ which implies $v_h|_T \in P_{m'}(T)$. Also, numerical stability requires positive weights and we refrain that the bilinear form a is elliptic. Then

$$\begin{aligned}
\tilde{a}(v_h) &= \sum_T Q_T \left(\sum_{\nu,\mu=1}^{2} a_{\nu\mu}\partial_\mu v_h\, \partial_\nu v_h\right) \\
&\overset{(128),(131)}{\geq} c\sum_T Q_T(|\nabla v_h|^2) \qquad \text{(uniform ellipticity of } a\text{)} \\
&\overset{(156)}{\geq} c\sum_T \int_T |\nabla v_h|^2\, dx \\
&= c|v_h|_1^2 \\
&\geq c||v_h||_1^2 \qquad \text{(Poincaré for functions of } H_0^1(\Omega)\text{)}
\end{aligned}$$

The last estimation follows from the equivalence of $||\cdot||_1$ and $|\cdot|_1$ on H_0^1.

The conditions of Lemma (8.141) are true and the proof is finished.

□

The previous theorem yields the order convergence of the FEM for numerical integration.

Corollary 8.150. *The conditions of Theorem* (8.149) *hold. Then, Lemma* (8.141) *leads to a quantitative error estimation, with* $\tau = r - m + 2$*,*

$$||u_h - \tilde{u}_h||_1 = \mathcal{O}(h^{r-m+2})$$

The total error is given by

$$||u - \tilde{u}_h||_1 \leq ||u - u_h||_1 + ||u_h - \tilde{u}_h||_1 = \mathcal{O}(h^{m-1} + h^{r-m+2})$$

Convergence is assured if $r \geq m - 1$*. In other words*

$$r' \geq m' \tag{165}$$

Remark that r' *denotes the degree of the quadrature rule and* m' *the degree of the finite elements.*

Example 8.151. *For* $r = m - 1$ *we have,*

$$||u - \tilde{u}_h||_1 = \mathcal{O}(h)$$

Optimal convergence is given if $r \geq 2m - 3$*. In example* $r = 2m - 3$*:*

$$||u - \tilde{u}_h||_1 = \mathcal{O}(h^{m-1})$$

The error of quadrature can be neglected if $r > 2m - 3$*, because*

$$||u - \tilde{u}_h||_1 = \mathcal{O}(h^{m-1})$$

is more important for the order of convergence.

Remark 8.152. *The inequality* (165) *is the main assertion of the report. But remind that* $r' >> 2m' - 1$ *(degree of quadrature rule very large in comparison to the optimal order of convergence) is not recommended because the calculation of the coefficient functions* $a_{ik}(x)$ *and the gradients* $\partial_\mu w_i \, \partial_\nu w_j, i, j = 1, \ldots N$*,* $\mu, \nu = 1, 2$ *could be very expensive for higher order quadrature rules.*

8.15.3.4 Concluding remarks The central result is the Lemma of Strang (8.141) which gives a quantitative error estimation for convergence and leads to Theorem (8.149).

The main condition is the uniform ellipticity of the approximate bi linear form $\tilde{a}$.

If the conditions are not satisfied it may be possible that the linear equation system $Au = b$ is not solvable because matrix A is singular. We study one example for this situation.

8.15.4 Numerical tests

8.15.4.1 Example 1 We discuss one simple example for a situation where the system matrix will be singular. Since $r' \geq m$ we have convergence of the procedure. Where r' means the degree of the quadrature rule and m is the degree of the FEM polynomials.

We assume the one-dimensional Poisson problem, take quadratic finite elements and use the middle-point rule as integration formula. So we solve

$$-u'' = f, \quad u(0) = (1) = 0$$

and take quadratic basic functions in the interval $[0, 1]$ with equidistant step size $h = x_i - x_{i-1}$. There is an additional point required, so we take the middle point of the cells T_i,

$$x_{i-1/2} := \frac{1}{2}(x_{i-1} - x_i)$$

The construction of the quadratic function v satisfies

$$v(x) = \sum_{i=0}^{N} v_i \psi_i(x) + \sum_{i=1}^{N} v_{i-1/2} \psi_{i-1/2}(x)$$

with the following properties

i) $\psi_i(x_k) = \delta_{ik}, \quad \psi_i(x_{k-1/2}) = 0$

ii) $\psi_{i-1/2}(x_k) = 0, \quad \psi_{i-1/2}(x_{k-1/2}) = \delta_{ik}$

Now, we get three quadratic basic functions

$$\psi_i(x) = \begin{cases} \frac{(x-x_{i-1})(x-x_{i-1/2})}{(x_i-x_{i-1})(x_i-x_{i-1/2})}, & x \in T_i \\ \frac{(x-x_{i+1})(x-x_{i+1/2})}{(x_i-x_{i+1})(x_i-x_{i+1/2})}, & x \in T_{i+1} \\ 0, & \text{otherwise} \end{cases}$$

and

$$\psi_{i-1/2}(x) = \begin{cases} \frac{(x-x_{i-1})(x-x_i)}{(x_{i-1/2}-x_i)(x_{i-1/2}-x_{i+1})}, & x \in T_i \\ 0, & \text{otherwise} \end{cases}$$

Next task is constructing the derivatives and give the variational formulation of the one-dimensional Laplace problem. Then solving integrals with middle-point rule,

$$\int_a^b f(x)\,dx \approx (b-a)\, f\left(\frac{a+b}{2}\right) + \frac{1}{24} f''(\xi)(b-a)^3 \tag{166}$$

One know that the middle-point rule has degree 1. Using this quadrature formula for quadratic elements fails and the resulting systemmatrix A will have some zeros, so it is singular. For a local element we obtain

$$\begin{aligned} \partial_x \psi_{i-1}(x) &= \frac{2}{h_i^2}(2x - x_{i-1/2} - x_i) \\ \partial_x \psi_{i-1/2}(x) &= \frac{2}{h_i^2}(x_i + x_{i-1} - 2x) \\ \partial_x \psi_i(x) &= \frac{2}{h_i^2}(2x - x_{i-1/2} - x_{i-1}) \end{aligned}$$

and the following integrals

$$\begin{aligned} A_{11} &= \int_{x_{i-1}}^{x_i} (\partial_x \psi_{i-1})^2\,dx \\ A_{22} &= \int_{x_{i-1}}^{x_i} (\partial_x \psi_{i-1/2})^2\,dx \approx h_i (\partial_x \psi_{i-1/2}(x_{i-1/2}))^2 = 0 \\ A_{33} &= \int_{x_{i-1}}^{x_i} (\partial_x \psi_i)^2\,dx \\ A_{12} = A_{21} &= \int_{x_{i-1}}^{x_i} \partial_x \psi_{i-1} \cdot \partial_x \psi_{i-1/2}\,dx \approx h_i \left[(\partial_x \psi_{i-1} \cdot \partial_x \psi_{i-1/2})(x_{i-1/2}) \right] = 0 \\ A_{13} = A_{31} &= \int_{x_{i-1}}^{x_i} \partial_x \psi_{i-1} \cdot \partial_x \psi_i\,dx \\ A_{23} = A_{32} &= \int_{x_{i-1}}^{x_i} \partial_x \psi_{i-1/2} \cdot \partial_x \psi_i\,dx \approx h_i \left[(\partial_x \psi_{i-1/2} \cdot \partial_x \psi_i)(x_{i-1/2}) \right] = 0 \end{aligned}$$

At last we write the results in our matrix,

$$A = \begin{pmatrix} A_{11} & 0 & A_{13} \\ 0 & 0 & 0 \\ A_{31} & 0 & A_{33} \end{pmatrix} \tag{167}$$

8.15.4.2 Example 2 The second example handles also with the Laplace equation but in 2D implemented again in deal.II [6]. Our focuss is on two lines in this program:

```
// m' : degree of the finite elements
fe (m'),
```

and

```
// Gauss formula, where n gives the number of notes in each direction
QGauss<2> quadrature_formula(n)
```

The degree of Gauss quadrature is given by $r' = 2n - 1$. We made some calculations for different m' and r'. The results are

n	1	2	3	4
r'	1	3	5	7
m'				
1	OK	OK	OK	OK
2	F	OK	OK	OK
3	F	ok	OK	OK
4	F	F	ok	OK
5	F	F	ok	ok
6	F	F	F	ok
7	F	F	F	ok
8	F	F	F	F

We shortly explain the notation:

- ok: (normal) convergence takes place
- OK: Optimal convergence is given with $r' \geq 2m - 1$
- F: no convergence

8.16 Numerical tests and computational convergence analysis

We finish this long chapter with several numerical tests in 2D and 3D.

8.16.1 2D Poisson

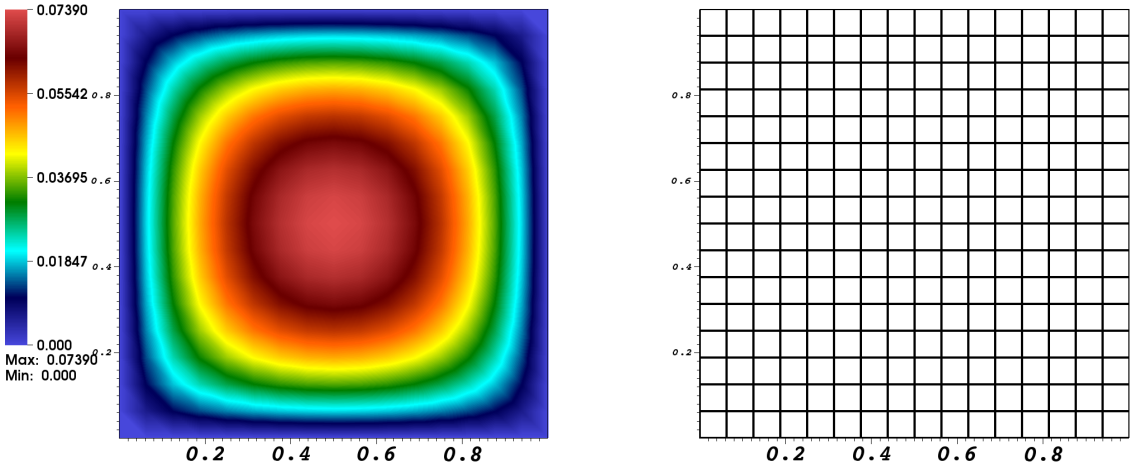


Figure 28: The graphical solution (left) and the corresponding mesh (right).

We compute Poisson in 2D on $\Omega = (0,1)^2$ and homogeneous Dirichlet conditions on $\partial\Omega$. The force is $f = 1$. We use a quadrilateral mesh using Q_1 elements (in an isoparametric FEM framework). The number of mesh elements is 256 and the number of DoFs is 289. We need 26 CG iterations (see Section 10.4 for the development of the CG scheme), without preconditioning, for the linear solver to converge.

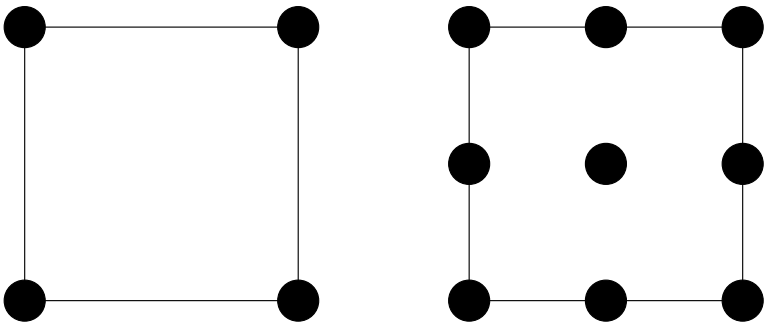


Figure 29: Q_1 (bilinear) and Q_2 (biquadratic) elements on a quadrilateral in 2D.

8.16.2 Numerical test: 3D

```
Number of elements:   4096
Number of DoFs:       4913
Number of iterations: 25 CG steps without preconditioning.
```

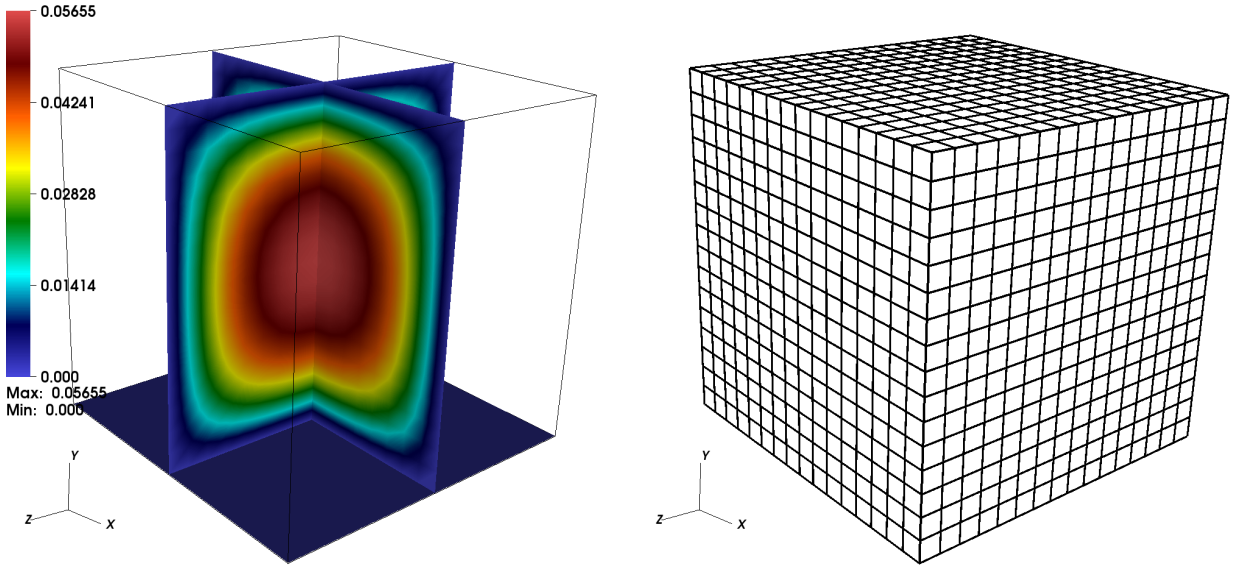


Figure 30: The graphical solution (left) and the corresponding mesh (right).

8.16.3 Checking programming code and convergence analysis for linear and quadratic FEM

We present a general algorithm and present afterwards 2D results.

Algorithm 8.153. *Given a PDE problem. E.g.* $-\Delta u = f$ *in* Ω *and* $u = 0$ *on the boundary* $\partial\Omega$.

1. *Construct by hand a solution* u *that fulfills the boundary conditions.*
2. *Insert* u *into the PDE to determine* f.
3. *Use that* f *in the finite element simulation to compute numerically* u_h.
4. *Compare* $u - u_h$ *in relevant norms (e.g.,* L^2, H^1*).*
5. *Check whether the desired* h *powers can be obtained for small* h.

We demonstrate the previous algorithm for a $2D$ case in $\Omega = (0, \pi)^2$:

$$\begin{aligned} -\Delta u(x, y) &= f \quad \text{in } \Omega, \\ u(x, y) &= 0 \quad \text{on } \partial\Omega, \end{aligned}$$

and we constuct $u(x, y) = \sin(x)\sin(y)$, which fulfills the boundary conditions (trivial to check! But please do it!). Next, we compute the right hand side f:

$$-\Delta u = -(\partial_{xx}u + \partial_{yy}u) = 2sin(x)sin(y) = f(x, y).$$

8.16.3.1 Numerically-obtained reference values On a seven-times refined mesh (16 384 elements), we obtain as point-value goal functionals using Q_c^1 elements:

$$u(0.5, 0.5) = 0.0736749...$$
$$u(0.75, 0.75) = 0.0452887...$$

For Q_c^2, we obtain

$$u(0.5, 0.5) = 0.0736714...$$
$$u(0.75, 0.75) = 0.0452862...$$

These values can be compared to the previously-constructed manufactured solution evaluated in these point values. Moreover, these values can be more easily taken into account to quickly check an own developed solver. For a sophisticated numerical analysis, we go now into the next paragraphs below.

8.16.3.2 2D Poisson: Linear FEM with Q_c^1 elements We then use f in the above program from Section 8.16.1 and evaluate the L^2 and H^1 norms using linear FEM. The results are:

Level	Elements	DoFs (N)	h	L2 err	H1 err
2	16	25	1.11072	0.0955104	0.510388
3	64	81	0.55536	0.0238811	0.252645
4	256	289	0.27768	0.00597095	0.126015
5	1024	1089	0.13884	0.00149279	0.0629697
6	4096	4225	0.06942	0.0003732	0.0314801
7	16384	16641	0.03471	9.33001e-05	0.0157395
8	65536	66049	0.017355	2.3325e-05	0.00786965
9	262144	263169	0.00867751	5.83126e-06	0.00393482
10	1048576	1050625	0.00433875	1.45782e-06	0.00196741
11	4194304	4198401	0.00216938	3.64448e-07	0.000983703

In this table we observe that we have quadratic convergence in the L^2 norm and linear convergence in the H^1 norm. For a precise (heuristic) computation, we also refer to Chapter 16 in which a formula for computing the convergence orders α is derived.

8.16.3.3 2D Poisson: Quadratic FEM with Q_c^2 elements We use again f in the above program from Section 8.16.1 and evaluate the L^2 and H^1 norms using quadratic FEM. The results are:

Level	Elements	DoFs (N)	h	L2 err	H1 err
2	16	81	1.11072	0.00505661	0.0511714
3	64	289	0.55536	0.000643595	0.0127748
4	256	1089	0.27768	8.07932e-05	0.00319225
5	1024	4225	0.13884	1.01098e-05	0.000797969
6	4096	16641	0.06942	1.26405e-06	0.000199486
7	16384	66049	0.03471	1.58017e-07	4.98712e-05
8	65536	263169	0.017355	1.97524e-08	1.24678e-05
9	262144	1050625	0.00867751	2.46907e-09	3.11694e-06
10	1048576	4198401	0.00433875	3.08687e-10	7.79235e-07
11	4194304	16785409	0.00216938	6.14696e-11	1.94809e-07

In this table we observe that we have cubic convergence $O(h^3)$ in the L^2 norm and quadratic convergence $O(h^2)$ in the H^1 norm. This confirms the theory; see for instance [24]. For a precise (heuristic) computation, we also refer to Chapter 16 in which a formula for computing the convergence orders α is derived.

We next plot the DoFs versus the errors in Figure 31 in order to highlight the convergence orders. For the relationship between h and DoFs versus the errors in different dimensions, we refer again to Chapter 16.

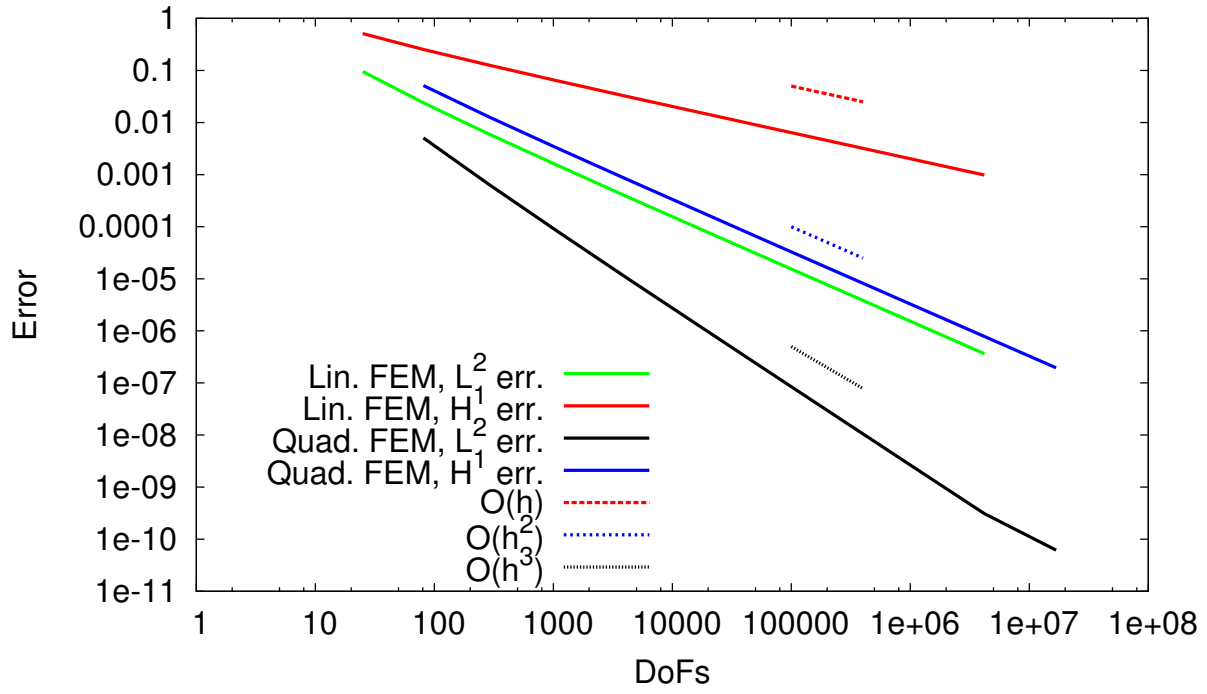


Figure 31: Plotting the DoFs versus the various errors for the 2D Poisson test using linear and quadratic FEM. We confirm numerically the theory: we observe $\|u-u_h\|_{L^2} = O(h^{r+1})$ and $\|u-u_h\|_{H^1} = O(h^r)$ for $r = 1, 2$, where r is the FEM degree.

8.16.4 Convergence analysis for 1D Poisson using linear FEM

We continue Section 8.4.9. As manufactured solution we use

$$u(x) = \frac{1}{2}(-x^2 + x),$$

on $\Omega = (0, 1)$ with $u(0) = u(1) = 0$, which we derived in Section 8.1.1. In the middle point, we have $u(0.5) = -0.125$ (in theory and simulations). In the following table, we plot the L^2 and H^1 error norms, i.e., $\|u - u_h\|_X$ with $X = L^2$ and $X = H^1$, respectively.

Level	Elements	DoFs (N)	h	L2 err	H1 err
1	2	3	0.5	0.0228218	0.146131
2	4	5	0.25	0.00570544	0.072394
3	8	9	0.125	0.00142636	0.0361126
4	16	17	0.0625	0.00035659	0.0180457
5	32	33	0.03125	8.91476e-05	0.00902154
6	64	65	0.015625	2.22869e-05	0.0045106
7	128	129	0.0078125	5.57172e-06	0.00225528
8	256	257	0.00390625	1.39293e-06	0.00112764
9	512	513	0.00195312	3.48233e-07	0.000563819
10	1024	1025	0.000976562	8.70582e-08	0.000281909

Using the formulae from Chapter 16, we compute the convergence order. Setting for instance for the L^2 error we have

$$P(h) = 1.39293e-06$$
$$P(h/2) = 3.48233e-07$$
$$P(h/4) = 8.70582e-08.$$

Then:

$$\alpha = \frac{1}{log(2)} log\Big(\Big|\frac{1.39293e-06 - 3.48233e-07}{3.48233e-07 - 8.70582e-08}\Big|\Big) = 1.99999696186957,$$

which is optimal convergence that confirm the a priori error estimates from Section 8.13.3. The octave code is:

```
alpha = 1 / log(2) * log(abs(1.39293e-06 - 3.48233e-07) / abs(3.48233e-07 - 8.70582e-08))
```

For the H^1 convergence order we obtain:

```
alpha = 1 / log(2) * log(abs(0.00112764 - 0.000563819) / abs(0.000563819 - 0.000281909))
      = 1.00000255878430,
```

which is again optimal convergence; we refer to Section 8.13.2 for the theory.

8.16.5 Computing the error norms $\|u - u_h\|$

To evaluate the global error norms $\|u - u_h\|_{L^2(\Omega)}$ and $\|u - u_h\|_{H^1(\Omega)}$, we proceed element-wise and define:

Definition 8.154 (See for example [24](Chapter 2, Section 6)). *For a decomposition $\mathcal{T}_h = \{K_1, \dots, K_M\}$ of the domain Ω and $m \geq 1$ we define*

$$\|u\|_{H^m(\Omega)} := \sqrt{\sum_{K_j} \|u\|^2_{H^m(K_j)}}.$$

Therefore we compute element-wise the error contributions:

$$\|u\|^2_{H^m(K_j)} = \int_{K_j} \sum_{|\alpha| \leq m} \|\partial^\alpha u\|^2_{L^2(K_j)},$$

where α is a multiindex representing the degree of the derivatives; see Section 3.6.

Example 8.155 (L^2 and H^1). *For L^2 it holds:*

$$\|u\|^2_{L^2(K_j)} = \int_{K_j} u^2 \, dx$$

For H^1 it holds:

$$\|u\|^2_{H^1(K_j)} = \int_{K_j} (u^2 + \nabla u^2) \, dx$$

Consequently for the errors:

Example 8.156 (Errors in L^2 and H^1). *For L^2 it holds:*

$$\|u - u_h\|^2_{L^2(K_j)} = \int_{K_j} (u - u_h)^2 \, dx$$

For H^1 it holds:

$$\|u - u_h\|^2_{H^1(K_j)} = \int_{K_j} ((u - u_h)^2 + (\nabla u - \nabla u_h)^2) \, dx,$$

which allow directly the evaluation of the integrals, or better, since u_h only lives in the discrete nodes (or quadrature points), we use directly a quadrature rule.

Then we obtain:

Proposition 8.157. *It holds:*

$$\|u-u_h\|_{L^2(\Omega)} = \sqrt{\sum_{K_j} \|u-u_h\|^2_{L^2(K_j)}},$$
$$\|u-u_h\|_{H^1(\Omega)} = \sqrt{\sum_{K_j} \|u-u_h\|^2_{H^1(K_j)}}.$$

8.17 Chapter summary and outlook

In this key chapter, we have introduced the theoretical background of finite elements, namely variational formulations. We discussed the properties and characteristic features of the finite element method in one dimension and higher dimensions. The discretization is based on uniform meshes (grids). For large-scaled problems and practical applications, uniform meshes are however unfeasible. In the next chapter, we introduce a posteriori error estimation that can be used to extract localized error indicators, which are then used for adaptive mesh refinement such that only 'important' parts of the numerical solution are resolved with a desired accuracy. Less important solution parts can be still treated with relatively coarse meshes. In Chapter 10, we discuss some numerical solution techniques such as the CG scheme, which has already been employed in the current chapter.

9 Goal-oriented a posteriori error estimation and adaptivity

This chapter is devoted to goal-oriented adaptivity based on a posteriori error estimation. The error estimation is carried out with the dual-weighted residual method (DWR) in which an adjoint problem must be solved. This method is particularly suited for problem in which general error functionals with only parts of the numerical solution u_h is of interest. In these cases, we need a high accuracy of the numerical solution, but uniform (i.e., global) mesh refinement is very expensive for large problems and three-dimensional computations. Here, memory requirements and CPU wall clock simulation times are too high to be justified. Mesh adaptivity tries to refine only a few mesh elements such that the envisaged discretization error reduces below a given tolerance. Simultaneously, mesh regions out of interest for the total accuracy may be coarsened. Similarly these concepts can be applied to the adaptive control of solution algorithms as for instance linear or nonlinear solvers.

9.1 Overview of the special class WS 19/20 on goal-oriented error estimation

1. Class 1: Motivation and principles: why error estimation, why adaptivity, examples, efficiency/robustness of error estimators, goal-functionals, Lagrangian (combining the PDE and the goal functional); see Section 9.2
2. Class 2: Errors and adjoints in $\mathbb{R}^n$ for linear equation systems [11][Section 1.4]
3. Class 3: DWR in time for ODEs (I) [11][Chapter 2]
4. Class 4: DWR in time for ODEs (II) [11][Chapter 2]; algorithm for end-time error estimation in which the initial value for the adjoint is numerically approximated
5. Class 5: Spatial DWR for linear Poisson (classical localization; integration by parts to obtain error information from neighboring elements) and approximation of the adjoint problem (Section 9.12.1)
6. Class 6: Spatial DWR for linear Poisson with PU localization 9.12.3; effectivity of the classical localization [113]
7. Class 7, Dec 3, 2019: Effectivity PU localization [113]; DWR for nonlinear spatial problems; adaptive algorithms; balancing discretization and nonlinear iteration error [45, 107]
8. Class 8, Dec 5, 2019: Ex 3; algorithms fixed-point and Newton for balancing discretization and nonlinear iteration error [45, 107]; two-sided error estimates [46] (Corr. 3.1, Lemma 3.2, Assumpt. 1, Theorem 3.3, Sec. 3.4 with Theorem 3.7, Part 1)
9. Class 9, Dec 12, 2019: Ex 4; Multiple goal functionals [47][Section 3]
10. Class 10, Dec 17, 2019: proof of enrichment for the combined goal functional [47][Prop. 3.1], adaptive algorithm for multiple goal functionals [47][Section 3.1]; practical aspects: refining elements (triangles, quads, prisms, hex; hanging nodes); refinement strategies: averaging, fixed-rate, fixed-fraction [11][Chapter 4]
11. Class 11, Jan 7, 2020: Space-time DWR [11][Chapter 9]; a primal-part error estimator for PU-DWR space-time in which $a(u)(\varphi)$ may be semi-linear
12. Class 12: Jan 14, 2020: Space-time DWR Besier (Schmich)/Vexler [119]. Further literature (not done): Rannacher [19] and [18]
13. Class 13, Jan 16, 2020: Space-time DWR Besier (Schmich)/Vexler [119]. Further literature (not done): Rannacher [19] and [18]
14. Class 14, Jan 28, 2020: Open questions, future possibilities, further applications

9.2 Motivation (Class 1)

Why adaptivity?

- Allows for flexible algorithms, efficient evaluations of numerical solutions or parts of it
- Decreasing the computational cost
- More important than ever because of big data, multipyisics applications, ...

How?

- Goal-oriented algorithms
- Approach motivated from optimization (Lagrange formalism)
- Adjoint equations

What?

- Different equations
- Related linear and nonlinear equations
- A posteriori error estimates
- Discretization error
- Nonlinear/linear iteration errors

Exercise 9. *Do you have examples of such goal functionals or parts of the solution? A: Displacements, flow, heat transfer, chemical reactions, adaptive choice of tolerances as stopping criteria in $Ax = b$ or nonlinear iterations such as Newton or fixed-point.*

9.2.1 Goal functionals in differential equations and solution systems

Examples of goal functionals in differential equations:

- Given the operator equation $A(y) = 0$ with $A(y) := y'(t) - ay(t)$. Goal: end time value $y(T)$ of for instance a population
- Mean value distribution
$$\int_{t_1}^{t_2} y(t)\,dt$$
- L^2 norms of point values in $-u''(x) = f$. For instance
$$\int_{x_1}^{x_2} u(x)\,dx$$
or
$$\left(\int_{x_1}^{x_2} u(x)^2\,dx\right)^{1/2}$$
or point values $u(x_l)$ or
$$\max_{x\in\Omega} |u(x)|$$
- Or in higher-order dimensions for $-\Delta u = f$ as well
$$\int_\Gamma \partial_n u\,ds$$

- Flow equations (Navier-Stokes)

$$\int_\Omega \nabla \times v \, dx$$

  or stresses

$$\int_\Gamma \sigma \cdot n \, ds$$

- Multiphysics: possibly multiple goal functionals
- Solvers (Newton): Solve $f(x) = 0$. Then, we need $|f(x_k)| < TOL$. Question: how to choose TOL? Depending on the iteration k, the tolerance TOL can be chosen adaptively. x_1 is far away from the solution, TOL may be still large. x_k is closer to the solution, TOL now small.

9.2.2 Abstract formulation of all these goal functionals

Definition 9.1 (Goal functional)**.** *A goal functional (quantity of physical interest; QoI) characterizes a part of a (or the entire) mathematical solution to a given mathematical problem.*

Definition 9.2 (Goal functional; math. definition)**.** *Let V be a function space. A goal functional is defined as a mapping*

$$J : V \to \mathbb{R}.$$

The result is simply a number in $\mathbb{R}$. Often, the error is measured in terms of this goal functional and here, the result characterizes the error in percentage.

Example 9.3. *We present some examples:*

1. *$J(y) = 0.01$ mean that the absolute error of y is measured in terms of J is 1 per cent.*
2. *$J(y) := y(T)$ is the end time error*
3. *$J(y) := \int_{t_1}^{t_2} y(t) \, dt$.*

9.2.3 Error, estimates, accuracy

Either way with goal functionals or usual norms, we are interested in the distance between numerical approximations and reference values. We recall briefly from Section 2.4. For instance, given $Ax = b$ and $A_h x_h = b_h$, the question is about the distance between x and x_h:

$$\|x - x_h\| = ? \quad \text{(Iteration error)}$$

Or for differential equations, given $-\Delta u = f$ and $-\Delta u_h = f$, what is

$$\|u - u_h\| = ? \quad \text{(Discretization error)}$$

To address these errors, we work with a priori or a posteriori error estimation; see Section 9.4.

Definition 9.4. *To estimate the previous errors, we work with an estimator η. See further in Section 9.5.*

9.3 Difference between norm-based error estimation and goal functionals

Goal functional error estimation is more general as norm-based error estimation. The main difference is that in norm-based error estimation we have

$$\|u - \tilde{u}\| = 0 \quad \Leftrightarrow \quad u - \tilde{u} = 0,$$

and $\|u - \tilde{u}\| \geq 0$. For goal functionals, it can be that

$$J(u) - J(\tilde{u}) = 0 \quad \text{for } u - \tilde{u} \neq 0$$

and in general $J(u) - J(\tilde{u}) \in \mathbb{R}$, but not necessarily positive.

9.4 Principles of error estimation

In general, we distinguish between **a priori** and **a posteriori error estimation**. In the first one, which we already have discussed for finite differences and finite elements, the (discretization) error is estimated before we start a numerical simulation/computation. Often, there are however constants c and (unknown) higher-order derivatives $|u|_{H^m}$ of the (unknown) solution:

$$\|u - u_h\| \leq ch^m |u|_{H^m}.$$

Such estimates yield

- the asymptotic information of the error for $h \to 0$.
- In particular, they provide the expected order of convergence, which can be adopted to verify programming codes and the findings of numerical simulations for (simple?!) model problems.

However, a priori estimates do not contain useful information during a computation. Specifically, the determination of better, reliable, adaptive step sizes h is nearly impossible (except for very simple cases).

On the other hand, a posteriori error estimation uses information of the already computed u_h during a simulation. Here, asymptotic information of u is not required:

$$\|u - u_h\| \leq c\eta(u_h)$$

where $\eta(u_h)$ is a computable quantity, because, as said, they work with the known discrete solution u_h. Such estimates cannot in general predict the asymptotic behavior, the reason for which a priori estimates remain important, but they can be evaluated during a computation with two main advantages:

- We can **control** the error during a computation;
- We can possibly **localize** the error estimator in order to obtain local error information on each element $K_i \in \mathcal{T}_h$. The elements with the highest error information can be decomposed into smaller elements in order to reduce the error.

9.5 Efficiency, reliability, basic adaptive scheme

Following Becker/Rannacher [14, 15], Bangerth/Rannacher [11], and Rannacher [104], we derive a posteriori error estimates not only for norms, but for a general differentiable functional of interest $J : V \to \mathbb{R}$. This allows in particular to estimate technical quantities arising from applications (mechanical and civil engineering, fluid dynamics, solid dynamics, etc.). Of course, norm-based error estimation, $\|u - u_h\|$ can be expressed in terms of $J(\cdot)$; for hints see Section 9.18.

When designing a posteriori error estimates, we are in principle interested in the following relationship:

$$C_1\eta \leq |J(u) - J(u_h)| \leq C_2\eta \tag{168}$$

where $\eta := \eta(u_h)$ is the **error estimator** and C_1, C_2 are positive constants. Moreover, $J(u) - J(u_h)$ is the **true error**.

Definition 9.5 (Efficiency and reliablity)**.** *A good estimator $\eta := \eta(u_h)$ should satisfy two bounds:*

1. *An error estimator η of the form* (168) *is called* ***efficient*** *when*

$$C_1\eta \leq |J(u) - J(u_h)|,$$

which means that the error estimator is bounded by the error itself.

2. *An error estimator η of the form* (168) *is called* ***reliable*** *when*

$$|J(u) - J(u_h)| \leq C_2\eta.$$

Here, the true error is bounded by the estimator.

Ideally, it should hold in the asymptotic limit:

$$\left| \frac{\eta}{J(u) - J(u_h)} \right| \to 1.$$

This yields the so-called effectivity index; see also Section 9.10.

Remark 9.6. *For general goal functionals $J(\cdot)$, it is much more simpler to derive **reliable** estimators rather than prooving their efficiency.*

Remark 9.7 (AFEM)**.** *Finite element frameworks working with a posteriori error estimators applied to local mesh adaptivity, are called **adaptive FEM**.*

Definition 9.8 (Basic algorithm of AFEM)**.** *The basic algorithm for AFEM is always the same:*

1. ***Solve** the PDE on the current mesh $\mathcal{T}_h$;*
2. ***Estimate** the error via a posteriori error estimation to obtain η;*
3. ***Mark** the elements by localizing the error estimator;*
4. ***Refine/coarsen** the elements with the highest/lowest error contributions using a certain refinement strategy.*

A prototype situation on three meshes is displayed in Figure 32.

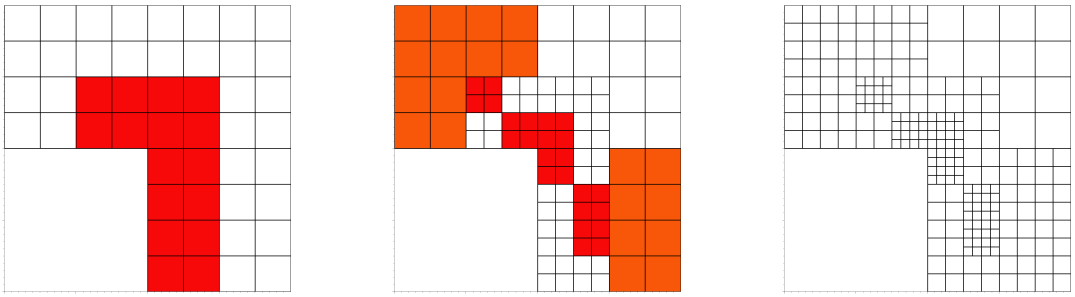


Figure 32: Meshes, say, on level 0, level 1 and 2. The colored mesh elements indicate high local errors and are marked for refinement using bisection and hanging nodes.

9.6 Goal-oriented error estimation using dual-weighted residuals (DWR)

9.6.1 Motivation

The goal of this section is to derive an error estimator that is based on duality arguments and can be used for norm-based error estimation (residual-based) as well as estimating more general error functionals $J(\cdot)$.

The key idea is based on numerical optimization, but goes back to classical mechanics in the 17th century. From our previous considerations, our goal is to reduce the error in the functional of interest with respect to a given PDE:

Formulation 9.9 (Goal functional evaluation as a constrained optimization problem)**.** *Let $u \in V$ the continuous (unknown) solution and $u_h \in V_h$ the numerical approximation. For a given PDE in weak from $a(u, \phi) = l(\phi)$ respectively, the discrete form $a(u_h, \phi_h) = l(\phi_h)$, we aim to minimize the goal functional error $J(u) - J(u_h)$. This can be written as a constrained optimization problem*

$$\min\big(J(u) - J(u_h)\big) \quad s.t. \quad a(u, \phi) = l(\phi), \tag{169}$$

where $J(\cdot)$ and $a(u, \phi)$ can be linear or nonlinear, but need to be differentiable (in Banach spaces).

Remark 9.10. *Of course for linear functionals, we can write*

$$J(u) - J(u_h) = J(u - u_h) = J(e), \quad e = u - u_h.$$

Such minimization problems as in (169) can be treated with the help of the so-called **Lagrangian** $L : V \times V \to \mathbb{R}$ in which the functional $J(\cdot)$ is of main interest subject to the constraint $a(\cdot,\cdot) - l(\phi)$ (here the PDE in variational form). Here, we deal with the **primal variable** $u \in V$ and a **Lagrange multiplier** $z \in V$, which is the so-called **adjoint variable** and which is assigned to the constraint $a(u,z) - l(z) = 0$. We then obtain

Definition 9.11. *The Lagrangian $L : V \times V \to \mathbb{R}$ is defined as*

$$L(u,z) = \big(J(u) - J(u_h)\big) - a(u,z) + l(z).$$

The Lagrange multiplier z measures the sensitivity of the PDE with respect to the given goal functional $J(u)$.

Before we proceed, we briefly recapitulate an important property of the Lagrangian. Let V and Z be two Banach spaces. A Lagrangian is a mapping $L(v,q) : U \times Y \to \mathbb{R}$, where $U \times Y \subset V \times Z$.

Definition 9.12 (Saddle-point)**.** *A point $(u,z) \in U \times Y$ is a saddle point of L on $U \times Y$ when*

$$\forall y \in Y : \quad L(u,y) \le L(u,z) \le L(v,z) \quad \forall v \in U.$$

A saddle-point is also known as min-max point. Geometrically one may think of a horse saddle.

Remark 9.13. *One can show that a saddle point yields under certain (strong) conditions a global minimum of u of $J(\cdot)$ in a subspace of U.*

The Lagrangian has the following important property to clear away the constraint (recall the constraint is the PDE!):

Lemma 9.14. *The problem*

$$\inf_{u \in V, -a(u,z)+l(z)=0} \big(J(u) - J(u_h)\big)$$

is equivalent to

$$\inf_{u \in V, -a(u,z)+l(z)=0} = \inf_{u \in V} \sup_{z \in V} L(u,z)$$

Proof. If $a(u,z) + l(z) = 0$ we clearly have $J(u) - J(u_h) = L(u,z)$ for all $z \in V$. If $a(u,z) + l(z) \neq 0$, then $\sup_{z \in V} L(u,z) = +\infty$, which shows the result. □

Remark 9.15. *In the previous lemma, we prefer to work with the infimum (inf) since it is not clear a priori whether the minimum (min) is taken.*

9.6.2 Excursus: Variational principles and Lagrange multipliers in mechanics

In this excursus we shall give a mechanics-based motivation of Lagrange multipliers and constrained optimization problems as they form the background of the current chapter. Specifically, we address a physical interpretation of **Lagrange multipliers**. We have motivated the Poisson problem as the clothesline problem in Section 6.1. The derivation based on first principles in physics, namely conservation of potential energy is as follows. Let a clothesline be subject to graviational forces g. We seek the solution curve $u = u(x)$. A final equilibrium is achieved for minimal potential energy. This condition can be expressed as:

Formulation 9.16 (Minimal potential energy - an unconstrained variational problem)**.** *Find u such that*

$$\min J(u)$$

with

$$J(u) = E_{pot} = \underbrace{\int_1^2 gu\, dm}_{\text{right hand side}} = \underbrace{\rho g \int_{x_1}^{x_2} u\sqrt{1 + (u')^2}\, dx}_{\text{left hand side}},$$

where g is the gravity, u the sought solution, dm mass elements, ρ the mass density. By variations δu we obtain solutions $u + \delta u$ and seek the optimal solution such that $J(u)$ is minimal. Of course, the boundary conditions u_1 and u_2 are not varied.

In the following, we formulate a constrained minimization problem. We ask that the length L of the clothesline is fixed:

$$K(u) = L = \int_{x_1}^{x_2} \sqrt{1 + (u')^2}\, dx = \text{const.}$$

Formulation 9.17 (A constrained minimization problem)**.** *Find u such that*

$$\min J(u) \quad s.t. \quad K(u) = const.$$

The question is how we address Formulation 9.17 in practice? We explain the derivation in terms of a 1D situation in order to provide a basic understanding as usually done in these lecture notes.

The task is:

$$\min J(x, u(x)) \quad \text{s.t.} \quad K(x, u(x)) = 0. \tag{170}$$

Let us assume for a moment, we can explicitly compute $u = u_K(x)$ from $K(x, u(x)) = 0$ such

$$K(x, u_K(x)) = 0.$$

The minimal value of $J(x, u)$ on the curve $u_K(x)$ can be computed as minimum of $J(x, u_K(x))$. For this reason, we use the first derivative to compute the stationary point. With the help of the chain rule, we obtain:

$$0 = \frac{d}{dx} J(x, u_K) = J'_x(x, u_K) + J'_u(x, u_K) u'_K(x). \tag{171}$$

With this equation, we obtain the solution x_1. The solution u_1 is then obtain from $u_1 = u_K(x_1)$.

Using **Lagrange multipliers**, we avoid the explicit construction of $u_K(x)$ because such an expression is only easy to obtain for simple model problems. To this end, we introduce the variable z as a Lagrange multiplier.

We build the Lagrangian

$$L(x, u, z) = J(x, u) - zK(x, u)$$

and consider the problem:

$$\min L(x, u, z) \quad \text{s.t. } K(x, u) = 0.$$

Again, to find the optimal points, we differentiate w.r.t. to the three solution variables x, u, z:

$$\begin{aligned} J'_x(x, u) - zK'_x(x, u) &= 0 \\ J'_u(x, u) - zK'_u(x, u) &= 0 \\ K(x, u) &= 0 \end{aligned} \tag{172}$$

to obtain a first-order optimality system for determining x, u, z.

Proposition 9.18. *The formulations* (171) *and* (172) *are equivalent.*

Proof. We show (172) yields (171). We assume that we know $u = u_K(x)$, but we do not need the explicit construction of that $u_K(x)$. Then, $K(x, u) = 0$ is equivalent to

$$K(x, u) = u - u_K(x) = 0.$$

We differentiate w.r.t. x and u and insert the resulting expressions into the first two equations in (172):

$$J'_x(x, u) + zu'_K(x) = 0, \tag{173}$$

$$J'_u(x, u) - z = 0. \tag{174}$$

The second condition is nothing else, but

$$z = J'_u(x, u).$$

Here, we easily see that the Lagrange multiplier measures the sensitivity (i.e., the variation) of the solution curve u with respect to the functional $J(x, u)$. Inserting $z = J'_u(x, u)$ into (172) yields (171). The backward direction can be shown in a similar fashion. □

Remark 9.19. *The previous derivation has been done for a 1D problem in which $u(x)$ with $x \in \mathbb{R}$ is unknown. The method can be extended to $\mathbb{R}$ and Banach spaces, the latter one being addressed in Section 9.6.3.*

Remark 9.20. *We emphasize again that the use of Lagrange multipliers seems more complicated, but avoids the explicit construction of $u_K(x)$, which can be cumbersome. This is the main reason of the big success of **adjoint methods**, working with the **adjoint variable** z in physics and numerical optimization. Again, here, the Lagrangian $L(x, u, z)$ is minimized and the solution $u = u(x, z)$ contains a parameter z. This parameter is automatically determined such that the constraint $K(x, u) = 0$ is satisfied. The prize to pay is a higher computational cost since more equations need to be solved using* (172) *in comparison to* (171).

9.6.3 First-order optimality system

As motivated in the previous subsections, we seek minimal points and therefore we look at the first-order necessary conditions. Now we work in Banach spaces rather than $\mathbb{R}$. Differentiation with respect to $u \in V$ and $z \in V$ yields the optimality system:

Proposition 9.21 (Optimality system: Primal and adjoint problems)**.** *Differentiating (see Section 7.4) the Lagrangian from Definition 9.11*

$$L(u, z) = \big(J(u) - J(u_h)\big) - a(u, z) + l(z),$$

yields:

$$\begin{aligned} L'_u(u, z)(\phi) &= J'_u(u)(\phi) - a'_u(u, z)(\phi) \quad \forall \phi \in V, \\ L'_z(u, z)(\psi) &= -a'_z(u, z)(\psi) + l'_z(\psi) \quad \forall \psi \in V. \end{aligned}$$

*The first equation is called the **adjoint problem** and the second equation is nothing else than our PDE, the so-called **primal problem**. We also observe that the trial and test functions switch in a natural way in the adjoint problem.*

Proof. Trivial with the methods presented in Section 7.4. □

Remark 9.22 (on the notation)**.** *We abuse a bit the standard notation for semi-linear forms. Usually, all linear and nonlinear arguments are distingished such that $a'_u(u, z)(\phi)$ would read $a'_u(u)(z, \phi)$ because u is nonlinear and z and ϕ are linear. We use in these notes however $a'_u(u, z)(\phi)$ in order to emphasize that u and z are the main variables.*

Corollary 9.23 (Primal and adjoint problems in the linear case)**.** *In the linear case, we obtain from the general formulation:*

$$\begin{aligned} L(\phi, z) &= J(\phi) - a(\phi, z) \\ L(u, \psi) &= -a(u, \psi) + l(\psi). \end{aligned}$$

Definition 9.24. *When we discretize both problems using for example a finite element scheme (later more), we define the residuals for $u_h \in V_h$ and $z_h \in V_h$. The primal and adjoint residuals read, respectively:*

$$\begin{aligned} \rho(u_h, \cdot) &= -a'_z(u_h, z)(\cdot) + l'_z(\cdot) \\ \rho^*(z_h, \cdot) &= J'_u(u)(\cdot) - a'_u(u, z_h)(\cdot). \end{aligned}$$

In the linear case:

$$\begin{aligned} \rho(u_h, \cdot) &= -a(u_h, \cdot) + l(\cdot) \\ \rho^*(z_h, \cdot) &= J(\cdot) - a(\cdot, z_h). \end{aligned}$$

Proposition 9.25 (First-order optimality system)**.** *To determine the optimal points $(u, z) \in V \times V$, we set the first-order optimality conditions to zero:*

$$\begin{aligned} 0 &= J'_u(u)(\phi) - a'_u(u)(z, \phi) \\ 0 &= -a'_z(u)(z, \psi) + l'_z(\psi). \end{aligned}$$

Here, we easily observe that the primal equation is nothing else than the bilinear form $a(\cdot,\cdot)$ we have worked with so far. The adjoint problem is new (well known in optimization though) and yields sensitivity measures z of the primal solution u with respect to the goal functional $J(\cdot)$.

Example 9.26 (Poisson problem)**.** *Let $a(u,\phi) = (\nabla u, \nabla\phi)$. Then: $a(u,z) = (\nabla u, \nabla z)$. Then,*

$$a'_u(u,z)(\phi) = (\nabla\phi, \nabla z),$$

and

$$a'_z(u,z)(\psi) = (\nabla u, \nabla\psi).$$

These derivatives are computed with the help of directional derivatives (Gâteaux derivatives) in Banach spaces.

9.6.4 Error estimation and adjoints in $\mathbb{R}^n$

In this section, we follow closely [11][Section 1.4]. Let $A, A_h \in \mathbb{R}^{n\times n}$ and vectors $b, b_h \in \mathbb{R}^n$, we consider $x, x_h \in \mathbb{R}^n$, which are obtained from solving the continuous and approximated problems

$$Ax = b, \quad A_h x_h = b_h.$$

As usually in these notes, h indicates the approximation (discretization) parameter. We recall from Numerik 1/2:

$$\begin{aligned} e &:= x - x_h \quad \text{(approximation error)} \\ \tau &:= A_h x - b_h \quad \text{(truncation error)} \\ \rho &:= b - Ax_h \quad \text{(residual)} \end{aligned}$$

A priori error analysis is based on the truncation error:

$$A_h e = A_h x - A_h x_h = A_h x - b_h = \tau,$$

yielding

$$e = \tau A_h^{-1}.$$

An error bound is obtained as follows:

$$\|e\| \le c_{S,h}\|\tau\|, \quad c_{S,h} := \|A_h^{-1}\|$$

where $c_{S,h}$ is the discrete stability constant. In general τ is not or difficult to compute since the (unknown) exact solution x enters. On the other hand, a posteriori error analysis is based on the residual as follows:

$$Ae = Ax - Ax_h = b - Ax_h = \rho.$$

From which we obtain

$$e = \rho A^{-1}$$

and thus

$$\|e\| \le c_S\|\rho\|, \quad c_S := \|A^{-1}\|.$$

The operator A is of course unknown, but may be available from analytical arguments or regularity theory.

9.6.5 Duality-based error estimation - linear problems

Let $j \in \mathbb{R}^n$ be given. We have

$$J(u) - J(u_h) = J(e) = (e, j)$$

The element j is determined by solving the adjoint problem:

$$A^* z = j.$$

Then, we obtain

$$J(e) = (e, j) = (e, A^* z) = (Ae, z) = (\rho, z),$$

which is an error relation e measured in the goal functional J by using the residual ρ and the adjoint solution z. Then:

$$|J(e)| = |(e, j)| = |(\rho, z)| \leq \|\rho\|\|z\| = \sum_{i=1}^{n} |\rho_i||z_i|.$$

The local residual is denoted by ρ_i and the weights z_i measure the influence of ρ_i on the target $J(u)$.

9.6.6 Duality-based error estimation - nonlinear problems

We assume differentiable mappings A and $A_h : \mathbb{R}^n \to \mathbb{R}^n$ and the vectors $b, b_h \in \mathbb{R}^n$. The problems are given by

$$A(x) = b, \quad A_h(x_h) = b_h.$$

Using the substitution rule, we have the following relation:

$$A(x) - A(x_h) = \int_0^1 A'(x_h + se)e \, ds, \quad e = x - x_h.$$

Using the scalar product, we then have

$$(A(x) - A(x_h), y) = \int_0^1 (A'(x_h + se)e, y) \, ds = (Be, y),$$

where

$$B := B(x, x_h) := \int_0^1 A'(x_h + se)e \, ds.$$

As in the linear case, we can define the adjoint problem

$$B^* z = j$$

yielding the error identity

$$J(e) = (e, j) = (e, B^* z) = (Be, z) = (A(x) - A(x_h), z) = (\rho, z)$$

and again

$$|J(e)| \leq := \sum_{i=1}^{n} |\rho_i||z_i|.$$

In the nonlinear case, a crucial difficulty is that the unknown error $e = x - x_h$ is included in the matrix B. One very simple possibility is

$$B \approx \tilde{B} = \int_0^1 A'(x_h) \, ds.$$

9.7 Duality-based error estimation for ODEs - adaptivity in time

See [11][Section 2].

9.8 Spatial error estimation for linear problems and linear goal functionals (Poisson)

We explain our developments in terms of the linear Poisson problem and linear goal functionals.

9.8.1 Primal problem

We work with

Formulation 9.27 (Primal problem)**.** *Let $f \in L^2(\Omega)$, and we assume that the problem and domain are sufficiently regular such that the trace theorem (see Theorem 8.82 and also the literature, e.g., [145]) holds true, i.e., $h \in H^{-\frac{1}{2}}(\Gamma_N)$, and finally $u_D \in H^{\frac{1}{2}}(\Omega)$. Find $u \in \{u_D + V\}$:*

$$a(u, \phi) = l(\phi) \quad \forall \phi \in V,$$

where

$$a(u, \phi) = (\alpha \nabla u, \nabla \phi)$$

and

$$l(\phi) := \int_\Omega f\phi \, dx + \int_{\Gamma_N} g\phi \, ds,$$

and the diffusion coefficient $\alpha := \alpha(x) \in L^\infty(\Omega)$. In this setting $\int_{\Gamma_N} g\phi \, ds$ has to be understood as duality product as for instance in [65]. If $g \in L^2(\Gamma_N)$ then it coincides with the integral.

9.8.2 Goal functional: some possible examples

As previously motivated, the aim is to compute a certain quantity of interest $J(u)$ with a desired accuracy at low computational cost.

Example 9.28. *Examples of goal functionals are mean values, line integration or point values:*

$$J(u) = \int_\Omega u \, dx, \quad J(u) = \int_\Gamma u \, ds, \quad J(u) = \int_\Gamma \partial_n u \, ds, \quad J(u) = u(x_0, y_0, z_0).$$

The first goal functional is simply the mean value of the solution. The third and fourth goal functionals are a priori not well defined. In case of the second functional we know the $\nabla u \in [L^2(\Omega)]^d$. Using the trace theorem (see Theorem 8.82), we can deduce that the trace in normal direction belongs to $H^{-\frac{1}{2}}(\partial\Omega)$. This however leads to the problem that the second functional is not always well defined. Concerning the third functional, we have previously shown in Section 7.1 that H^1 functions with dimension $d > 1$ the solution u is not any more continuous and the last evaluation is not well defined. If the domain and boundaries are sufficiently regular in 2D, the resulting solution is however H^2 regular and thanks to Sobolev embedding theorems (e.g., [34, 50]) also continuous.

Example 9.29. *Let $J(u) = \int_\Omega u \, dx$. Then the Fréchet derivative is given by*

$$J(\phi) = J'_u(u)(\phi) = \int_\Omega \phi \, dx$$

and, of course $J'_\lambda(u)(\psi) \equiv 0$.

The above goal functionals are however computed with a numerical method leading to a discrete version $J(u_h)$. Thus the key goal is to control the error $J(e) := J(u) - J(u_h)$ in terms of local residuals, which are computable on each mesh cell $K_i \in \mathcal{T}_h$.

9.8.3 Adjoint problem

According to the previous sections, we have

Proposition 9.30 (Adjoint problem)**.** *Based on the optimality system, here Corollary 9.23, we seek the adjoint variable $z \in V$:*

$$a(\phi, z) = J(\phi) \quad \forall \phi \in V. \tag{175}$$

Specifically, the adjoint bilinear form for the Poisson problem is given by

$$a(\phi, z) = (\alpha \nabla \phi, \nabla z).$$

For symmetric problems, the adjoint bilinear form $a(\cdot, \cdot)$ is the same as the original one, but differs for non-symmetric problems like transport for example.

Proof. Apply the first-order necessary condition. □

Remark 9.31. *Existence and uniqueness of this adjoint solution follows by standard arguments provided sufficient regularity of the goal functional and the domain are given. The regularity of $z \in V$ depends on the regularity of the functional J. For $J \in H^{-1}(\Omega)$ it holds $z \in H^1(\Omega)$. Given a more regular functional like the L^2-error $J(\phi) = \|e\|^{-1}(e_h, \phi)$ (where $e := u - u_h$) with $J \in L^2(\Omega)^*$ (denoting the dual space), it holds $z \in H^2(\Omega)$ on suitable domains (convex polygonal or smooth boundary with C^2-parameterization).*

9.8.4 Derivation of an error identity

We now work with the techniques as adopted in Section 8.13. Inserting as special test function $\psi := u - u_h \in V$, recall that $u_h \in V_h \subset V$ into (175) yields:

$$J(u - u_h) = a(u - u_h, z),$$

and therefore we have now a representation for the error in the goal functional.

Next, we use the Galerkin orthogonality $a(u - u_h, \psi_h) = 0$ for all $\psi_h \in V_h$, and we obtain (compare to Aubin-Nitsche in Section 8.13.3):

$$a(u - u_h, z) = a(u - u_h, z) - \underbrace{a(u - u_h, \psi_h)}_{=0} = a(u - u_h, z - \psi_h) = J(u - u_h). \tag{176}$$

The previous step allows us to choose ψ_h in such a way that $z - \psi_h$ can be bounded using interpolation estimates. Indeed, since ψ_h is an arbitrary discrete test function, we can for example use a projection $\psi_h := i_h z \in V_h$ in (176), which is for instance the nodal interpolation.

Definition 9.32 (Error identity)**.** *Choosing $\psi_h := i_h z \in V_h$ in* (176) *yields:*

$$J(u - u_h) = a(u - u_h, z - i_h z). \tag{177}$$

Thus, the error in the functional $J(u - u_h)$ can be expressed in terms of a residual, that is weighted by adjoint sensitivity information $z - i_h z$.

However, since $z \in V$ is an unknown itself, we cannot yet simply evaluate the error identity because z is only known analytically in very special cases. In general, z is evaluated with the help of a finite element approximation yielding $z_h \in V_h$.

However, this yields another difficulty since we inserted the interpolation $i_h : V \to V_h$ for $z \in V$. When we approximate now z by z_h and use a linear or bilinear approximation $r = 1$: $z_h \in V_h^{(1)}$, then the interpolation i_h does nothing (in fact we interpolate a linear/bilinear function z_h with a linear/bilinear function $i_h z_h$, which is clearly

$$z_h - i_h z_h \equiv 0.$$

For this reason, we need to approximate z_h with a scheme that results in a higher-order representation: here at least something of quadratic order: $z_h \in V_h^{(2)}$.

9.9 Nonlinear problems and nonlinear goal functionals

In the nonlinear case, the PDE may be nonlinear (e.g., p-Laplace, nonlinear elasticity, Navier-Stokes, and Section 4.6) and also the goal functional may be nonlinear, e.g.,

$$J(u) = \int_\Omega u^2 \, dx.$$

These nonlinear problems yield a semi-linear form $a(u)(\phi)$ (not bilinear any more), which is nonlinear in the first variable u and linear in the test function ϕ. Both the semi-linear form and the goal functional $J(\cdot)$ are assumed to be (Fréchet) differentiable.

We start from Definition 9.21. Assuming that we have discretized both problems using finite elements (for further hints on the adjoint solution see Section 9.9.1). We suppose that all problems have unique solutions. We define:

Definition 9.33 (Primal and adjoint residuals)**.** *We define the primal and adjoint residuals, respectively:*

$$\begin{aligned} \rho(u_h)(\phi) &= l(\phi) - a(u_h)(\phi) \quad \forall \phi \in V, \\ \rho^*(z_h)(\phi) &= J'_u(u_h)(\phi) - a'_u(u_h)(\phi, z) \quad \forall \phi \in V. \end{aligned}$$

It holds:

Theorem 9.34 ([15])**.** *For the Galerkin approximation of the first-order necessary system Definition 9.21, we have the* ***combined*** *a posteriori error representation:*

$$J(u) - J(u_h) = \eta = \frac{1}{2} \min_{\phi_h \in V_h} \rho(u_h)(z - \phi_h) + \frac{1}{2} \min_{\phi_h \in V_h} \rho^*(z_h)(u - \phi_h) + R.$$

The remainder term is of third order in $J(\cdot)$ and second order in $a(\cdot)(\cdot)$. Thus, for linear $a(\cdot)(\cdot)$ and quadratic $J(\cdot)$ the remainder term R vanishes. In practice the remainder term is neglected anyway and assumed to be small. However, in general this assumption should be justified for each nonlinear problem.

Proof. We refer the reader to [15]. □

Corollary 9.35 (Linear problems)**.** *In the case of linear problems the two residuals coincide. Then, it is sufficient to only solve the primal residual:*

$$\begin{aligned} J(u) - J(u_h) = J(u - u_h) &= \underbrace{\min_{\phi_h \in V_h} \rho(u_h)(z - \phi_h)}_{Primal} \\ &= \underbrace{\min_{\phi_h \in V_h} \rho^*(z_h)(u - \phi_h)}_{Adjoint} \\ &= \underbrace{\frac{1}{2} \min_{\phi_h \in V_h} \rho(u_h)(z - \phi_h) + \frac{1}{2} \min_{\phi_h \in V_h} \rho^*(z_h)(u - \phi_h)}_{Combined} \end{aligned}$$

Indeed the errors are exactly the same for linear goal functionals using the primal, adjoint or combined error estimator. The only difference are the resulting meshes. Several examples have been shown in Example 2 in [113].

Proof. It holds for the adjoint problem in the linear case:

$$J(\phi) = a(\phi, z).$$

Then:

$$\begin{aligned} J(u-u_h) = J(e) = \quad & \underbrace{a(u-u_h, z)}_{= l(z) - a(u_h, z)} \\ & \underbrace{\qquad\qquad}_{=\rho(u_h, z)} \\ & = a(e,z) = a(e, z-z_h) = a(e, e^*) = a(u-u_h, e^*) \\ & = \qquad\qquad \underbrace{a(u, e^*)}_{=a(u,z)-a(u,z_h)=\underbrace{J(u) - a(u, z_h)}_{=\rho^*(z_h, u)}} \\ & = J(e^*), \end{aligned}$$

where $e^* := z - z_h$ and where $a(e,z) = a(e, z-z_h)$ and $a(u-u_h, e^*) = a(u, e^*)$ hold true thanks to Galerkin orthogonality. □

Definition 9.36 (A formal procedure to derive the adjoint problem for nonlinear equations)**.** *We summarize the previous developments. Based on Proposition 9.21, we set-up the adjoint problem as follows:*

1. *Given a PDE plus boundary conditions, and in time-dependent cases initial conditions*
2. *Derive weak form $a(u)(\phi)$*
3. *Differentiate w.r.t. $u \in V$ such that*
$$a'_u(u)(\delta u, \phi)$$
with the direction $\delta u \in V$.
4. *Switch the trial function $\delta u \in V$ and the test function $\phi \in V$ such that:*
$$a'_u(u)(\phi, \delta u)$$
5. *Replace $\delta u \in V$ by the adjoint solution $z \in V$:*
$$a'_u(u)(\phi, z)$$
6. *This procedure also shows that the primal variable $u \in V$ enters the adjoint problem in the nonlinear case. However $u \in V$ is now given data and $z \in V$ is the sought unknown. This also means that in nonlinear problems, that primal solution needs to be stored in order to be accessed when solving the adjoint problem.*
7. *Construct error identity*
8. *Construct error estimator with $u \approx u_h^2$ and $z \approx z_h^2$*
9. *Possible localization (e.g., spatial refinement) of error estimator*

9.9.1 Approximation of the adjoint solution for the primal estimator $\rho(u_h)(\cdot)$

In order to obtain a computable error representation, $z \in V$ is approximated through a finite element function $z_h \in V_h$, that is obtained from solving a discrete adjoint problem:

Formulation 9.37 (Discrete adjoint problem for linear problems)**.**

$$a(\psi_h, z_h) = J(\psi_h) \quad \forall \psi_h \in V_h. \tag{178}$$

Then the primal part of the error estimator reads:

$$a(u-u_h, z_h - i_h z_h) = \rho(u_h)(z_h - i_h z_h) \approx J(u) - J(u_h). \tag{179}$$

The difficulty is that if we compute the adjoint problem with the same polynomial degree as the primal problem, then $z_h - i_h z_h \equiv 0$, and thus the whole error identity defined in (179) would vanish, i.e., $J(u) - J(u_h) \equiv 0$. This is clear from a theoretical standpoint and can be easily verified in numerical computations.

In fact normally an interpolation operator i_h interpolates from infinite dimensional spaces V into finite-dimensional spaces V_h or from higher-order spaces into low-order spaces.

Thus:

$$\begin{aligned} &i_h : V \to V_h^{(1)} && \text{so far...} \\ &i_h : V_h^{(1)} \to V_h^{(1)} && \text{first choice, but trivial solution, useless} \\ &i_h : V_h^{(2)} \to V_h^{(1)} && \text{useful choice with nontrivial solution} \end{aligned}$$

From these considerations it is also clear that

$$i_h : V_h^{(1)} \to V_h^{(2)}$$

will even be worse (this would arise if we approximate the primal solution with $u_h \in V_h^{(2)}$ and $z_h \in V_h^{(1)}$).

9.9.2 Summary and alternatives for computing the adjoint

In summary:

- the adjoint solution needs to be computed either with a global higher-order approximation (using a higher order finite element of degree $r+1$ when the primal problem is approximated with degree r),
- or a solution on a finer mesh,
- or local higher-order approximations using a patch-wise higher-order interpolation [11, 15, 113].

Clearly, the last possibility is the cheapest from the computational cost point of view, but needs some efforts to be implemented. For the convenience of the reader we tacitly work with a global higher-order approximation in the rest of this chapter.

We finally end up with the primal error estimator:

Definition 9.38 (Primal error estimator)**.** *The primal error estimator is given by:*

$$J(u) - J(u_h) =: \eta \approx a(u-u_h, z_h^{(r+1)} - i_h z_h^{(r+1)}) = \rho(u_h)(z_h^{(r+1)} - i_h z_h^{(r+1)}).$$

9.9.3 Approximation of the primal solution for the adjoint estimator $\rho^*(z_h)(\cdot)$

To evaluate the adjoint estimator $\rho(z_h)(\cdot)$, we need to construct

$$u - i_h u$$

with $i_h : V \to V_h$ for $u \in V$ and $u_h \in V_h$. Here we encounter the opposite problem to the previous section. We need to solve the primal problem with higher accuracy using polynomials of degree $r+1$ in order to construct a useful interpolation, yielding

$$u - i_h u \neq 0 \quad \text{a.e.}$$

Then:

$$J(u) - J(u_h) \approx \eta := \rho^*(z_h)(u_h^{(r+1)} - i_h u_h^{(r+1)}).$$

9.10 Measuring the quality of the error estimator η

As quality measure how well the estimator approximates the true error, we use the effectivity index I_{eff}:

Definition 9.39 (Effectivity index)**.** *Let η be an error estimator and $J(u) - J(u_h)$ the true error. The effectivity index is defined as:*

$$I_{eff} := I_{eff}(u_h, z_h) = \Big| \frac{\eta}{J(u) - J(u_h)} \Big|. \tag{180}$$

Problems with good I_{eff} satisfy asymptotically $I_{eff} \to 1$ for $h \to 0$. We say that

- *for $I_{eff} > 1$, we have an **over estimation** of the error,*
- *for $I_{eff} < 1$, we have an **under estimation** of the error.*

9.11 Rough structure of adjoint-based error estimation

We obtain as basic scheme:

1. Compute primal solution $\tilde{u}$
2. Compute adjoint solution $\tilde{z}$. For linear problems, both solutions are independent. For nonlinear PDEs, the primal solution $\tilde{u}$ enters into the adjoint problem. The latter follows naturally from the Lagrangian formulation.
3. Having just computed $\tilde{u}$ and $\tilde{z}$ construct error estimator $\eta(\tilde{u}, \tilde{z})$
4. Evaluate $\eta(\tilde{u}, \tilde{z})$ and compare with true error e. The latter of course can only be constructed as long as we deal with prototype problems for which we know the exact solution u.
5. Evaluate the quality of η by observing the effectivity index I_{eff}.

Using finer meshes (in order to obtain a higher approximation quality) for computing $\tilde{u}$ and $\tilde{z}$, we then repeat all previous steps and compare the different I_{eff}. The higher the approximation quality, the better (hopefully!!!) I_{eff} becomes. This procedure can be carried out for both uniform mesh refinement and adaptive mesh refinement. The latter is going to be explained in the following sections.

9.12 Localization techniques

In the previous sections, we have derived an error approximation η with the help of duality arguments that lives on the entire domain Ω. In order to use the error estimator for mesh refinement we need to localize the error estimator η to single mesh elements $K_j \in \mathcal{T}_h$ or degrees of freedom (DoF) i. We present two techniques:

- A classical procedure using integration by parts resulting in an element-based indicator η_K;
- A more recent way by employing a partition-of-unity (PU) yielding PU-DoF-based indicators η_i.

We recall that the primal estimator starts with $z \in V$ from:

$$\rho(u_h)(z - i_h z) = a(u - u_h, z - i_h z) = a(u, z - i_h z) - a(u_h, z - i_h z) = l(z - i_h z) - a(u_h, z - i_h z),$$

and the adjoint estimator

$$\rho^*(z_h)(u - i_h u) = J(u - i_h u) - a(u - i_h u, z).$$

According to Theorem 9.34, we have for linear problems

$$\begin{aligned} J(u) - J(u_h) &= \frac{1}{2}\rho + \frac{1}{2}\rho^* + R \\ &= \frac{1}{2}\big(l(z - i_h z) - a(u_h, z - i_h z)\big) + \frac{1}{2}\big(J'(u - i_h u) - a'(u - i_h u, z_h)\big) + R \end{aligned}$$

where R is the remainder term. Furthermore on the discrete level, we have (for linear problems):

$$J(u) - J(u_h) \approx \eta := \frac{1}{2}\big(l(z_h - i_h z_h) - a(u_h, z_h - i_h z_h)\big) + \frac{1}{2}\big(J(u_h - i_h u_h) - a(u_h - i_h u_h, z_h)\big). \tag{181}$$

We understand that the discrete solutions z_h in the first part and u_h in the second part have to be understood computed in terms of higher-order approximations $r+1$.

In the following we localize these error estimators on a single element $K_i \in \mathcal{T}_h$. Here, the influence of neighboring elements $K_j, j \neq i$ is important [32]. In order to achieve such an influence, we consider the error estimator on each cell and then either integrate back into the strong form (the classical way) or keep the weak form and introduce a partition-of-unity (a more recent way). Traditionally, there is also another way with weak form, proposed in [23], which has been analyzed theoretically in [113], but which we do not follow in these notes further.

In the following both localization techniques we present, we start from:

$$J(u - u_h) = l(z_h - i_h z_h) - a(u_h, z_h - i_h z_h) \tag{182}$$

For the Poisson problem, we can specify as follows:

$$J(u - u_h) = (f, z_h - i_h z_h) - (\nabla u_h, \nabla(z_h - i_h z_h)) \tag{183}$$

The next step will be to localize both terms either on a cell $K \in \mathcal{T}_h$ (Section 9.12.1) or on a degree of freedom (Section 9.12.3).

9.12.1 The classical way of error localization of the primal estimator for linear problems

In the classical way, the error identity (177) is treated with integration by parts on every mesh element $K \in \mathcal{T}_h$, which yields:

Proposition 9.40. *It holds:*

$$J(u - u_h) \approx \eta = \sum_{K \in \mathcal{T}_h} (f + \nabla \cdot (\alpha \nabla u_h), z_h - i_h z_h)_K + (\alpha \partial_n u_h, z_h - i_h z_h)_{\partial K} \tag{184}$$

Proof. Idea: Take (181), insert specific form of the PDE, reduce to an element K, integrate by parts. Now in detail: Let $\alpha = 1$ for simplicity. We start from

$$J(u - u_h) = (f, z_h - i_h z_h) - (\nabla u_h, \nabla(z_h - i_h z_h))$$

and obtain further

$$\begin{aligned}
J(u - u_h) &= (f, z_h - i_h z_h) - (\nabla u_h, \nabla(z_h - i_h z_h)) \\
&= \sum_K (f, z_h - i_h z_h)_K - (\nabla u_h, \nabla(z_h - i_h z_h))_K \\
&= \sum_K (f, z_h - i_h z_h)_K + (\Delta u_h, z_h - i_h z_h)_K - (\partial_n u_h, z_h - i_h z_h)_{\partial K} \\
&= \sum_K (f + \Delta u_h, z_h - i_h z_h)_K - \frac{1}{2}([\partial_n u_h]_K, z_h - i_h z_h)_{\partial K}
\end{aligned}$$

with $[\partial_n u_h] := [\partial_n u_h]_K = \partial_n u_h|_K + \partial_{n'} u_h|_{K'}$ where K' is a neighbor cell of K. On the outer (Dirichlet) boundary we set $[\partial_n u_h]_{\partial\Omega} = 2\partial_n u_h$. □

With the notation from the proof, we can define local residuals:

$$\begin{aligned}
R_T &:= f + \Delta u_h, \\
r_{\partial K} &:= -[\partial_n u_h]
\end{aligned}$$

Here, R_T are **element residuals** that measures the 'correctness' of the PDE. The $r_{\partial K}$ are so-called **face residuals** that compute the jumps over element faces and consequently measure the smoothness of the discrete solution u_h.

Further estimates are obtained as follows:

$$|J(u-u_h)| \leq \Big| \sum_{K\in\mathcal{T}_h} ... \Big| \leq \sum_{K\in\mathcal{T}_h} |...|$$

where we assume $z_h - \phi_h = 0$ on $\partial\Omega$. With Cauchy-Schwarz we further obtain:

$$\begin{aligned} |J(u-u_h)| \leq \eta &= \sum_K \Big[\|f + \Delta u_h\|_K \|z_h - i_h z_h\|_K + \frac{1}{2} \|[\partial_n u_h]\|_{\partial K \setminus \partial\Omega} \|z_h - i_h z_h\|_{\partial K} \Big] \\ &= \sum_K \rho_K(u_h)\omega_K(z_h) + \rho_{\partial K}(u_h)\omega_{\partial K}(z_h). \end{aligned}$$

In summary we have shown:

Proposition 9.41. *We have:*

$$|J(u) - J(u_h)| \leq \eta := \sum_{K\in\mathcal{T}_h} \rho_K \omega_K, \tag{185}$$

with

$$\rho_K := \|f + \nabla\cdot(\alpha\nabla u_h)\|_K + \frac{1}{2} h_K^{-\frac{1}{2}} \|[\alpha\partial_n u_h]\|_{\partial K}, \tag{186}$$

$$\omega_K := \|z - i_h z\|_K + h_k^{\frac{1}{2}} \|z - i_h z\|_{\partial K}, \tag{187}$$

where by $[\alpha\partial_n u_h]$ we denote the jump of the u_h derivative in normal direction. The residual part ρ_K only contains the discrete solution u_h and the problem data. On Dirichlet boundaries Γ_D, we set $[\alpha\partial_n u_h] = 0$ and on the Neumann part we evaluate $\alpha\partial_n u_h = g_N$. Of course, we implicitly assume here that $g_N \in L^2(\Gamma_N)$ such that these terms are well-defined.

Remark 9.42. *In practice, this primal error estimator needs to be evaluated in the dual space. Here, we proceed as follows:*

- *Prolongate the primal solution u_h into the dual space;*
- *Next, we compute the interpolation $i_h z_h^{(r+1)} \in Q_r$ w.r.t. to the primal space;*
- *Then, we compute $z_h^{(r+1)} - i_h z_h^{(r+1)}$ (here, $i_h z_h^{(r+1)}$ is prolongated to Q_{r+1} in order to compute the difference);*
- *Evaluate the duality product $\langle\cdot,\cdot\rangle$ and face terms.*

Remark 9.43. *When $V_h = V_h^{(1)}$, then $\nabla\cdot\nabla u_h \equiv 0$. This also demonstrates heuristically that face terms are important.*

9.12.2 The classical way for the combined estimator

The combined estimator reads:

Proposition 9.44. *It holds:*

$$J(u) - J(u_h) \approx \sum_{K\in\mathcal{T}_h} \frac{1}{2}\eta_K + \frac{1}{2}\eta_K^*$$

with

$$\begin{aligned} \eta_K &= \big(\langle f + \nabla\cdot(\alpha\nabla u_h), z_h - i_h z_h\rangle_K + \int_{\partial K} \alpha\partial_n u_h \cdot (z_h - i_h z_h)\, ds \big) \\ \eta_K^* &= \big(J(u_h - i_h u_h) - (\int_K ... + \int_{\partial K} ...) \big) \end{aligned}$$

9.12.3 A variational primal-based error estimator with PU localization

An alternative way is an DoF-based estimator, which is the first difference to before. The second difference to the classical approach is that we continue to work in the variational form and do not integrate back into the strong form. Such an estimator has been developed and analyzed in [113]. This idea combines the simplicity of the approach proposed in [23] (as it is given in terms of variational residuals) in terms of a very simple structure, which makes it particularly interesting for coupled and nonlinear PDE systems (see further comments below). Variational localizations are useful for nonlinear and coupled problems as we do not need to derive the strong form.

To this end we need to introduce a partition-of-unity (PU), which can be realized in terms of another finite element function. The procedure is therefore easy to realize in existing codes.

Definition 9.45 (PU - partition-of-unity)**.** *The PU is given by:*

$$V_{PU} := \{\psi_1, \dots, \psi_M\}$$

with $dim(V_{PU}) = M$*. The PU has the property*

$$\sum_{i=1}^{M} \psi_i \equiv 1.$$

Remark 9.46. *The PU can be simply chosen as the lowest order finite element space with linear or bilinear elements, i.e.,*

$$V_{PU} = V_h^{(1)}.$$

To understand the idea, we recall that in the classical error estimator the face terms are essential since they gather information from neighboring cells. When we work with the variational form, no integration by parts (fortunately!) is necessary. Therefore, the information of the neighboring cells is missing. Using the PU, we touch different cells per PU-node and consequently we gather now information from neighboring cells. Therefore, the PU serves as localization technique.

In the following, we now describe how the PU enters into the global error identity (177):

Proposition 9.47 (Primal error estimator)**.** *For the finite element approximation of Formulation 9.27, we have the a posteriori error estimate:*

$$|J(u) - J(u_h)| \leq \eta := \sum_{i=1}^{M} |\eta_i|, \tag{188}$$

where

$$\eta_i = a(u - u_h, (z - i_h z)\psi_i) = l((z - i_h z)\psi_i) - a(u_h, (z - i_h z)\psi_i),$$

and more specifically for the Poisson problem:

$$\eta_i = \Big\{ \langle f, (z - i_h z)\psi_i \rangle - (\alpha \nabla u_h, \nabla (z - i_h z)\psi_i) \Big\}. \tag{189}$$

9.12.4 PU localization for the combined estimator

Proposition 9.48 (The combined primal-dual error estimator)**.** *The combined estimator reads:*

$$|J(u) - J(u_h)| \leq \eta := \sum_{i=1}^{M} \frac{1}{2}|\eta_i| + \frac{1}{2}|\eta_i^*|$$

with

$$\begin{aligned} \eta_i &= \big(l((z_h - i_h z_h) - a(u, (z_h - i_h z_h)\psi_i)\psi_i) \big) \\ \eta_i^* &= \big(J_u'((u_h - i_h u_h)\psi_i) - a_u'((u_h - i_h u_h)\psi_i, z_h) \big) \end{aligned}$$

9.13 Comments to adjoint-based error estimation

Adjoint-based error estimation allows to measure precisely at a low computational cost specific functionals of interest $J(u)$. However, the prize to pay is:

- We must compute a second solution $z \in V$.
- This second solution inside the primal estimator must be of higher order, which means more computational cost in comparison to the primal problem.
- For the full error estimator in total we need to compute four problems.
- From the theoretical point of view, we cannot proof convergence of the adaptive scheme for general goal functionals.

For nonlinear problems, one has to say that the primal problem is subject to nonlinear iterations, but the adjoint problem is always a linearized problem. Here, the computational cost may become less significant of computing an additional adjoint problem. Nonetheless, there is no free lunch.

9.14 Balancing discretization and iteration errors

9.14.1 Main theorem

We recall from [45, 107] for the Galerkin case ($U = V$), we derive an error representation in the following theorem:

Theorem 9.49. *Let us assume that $\mathcal{A} \in \mathcal{C}^3(U, V)$ and $J \in \mathcal{C}^3(U, \mathbb{R})$. If u solves the primal problem and z solves the adjoint problem for $u \in U$, then it holds for arbitrary fixed $\tilde{u} \in U$ and $\tilde{z} \in V$:*

$$J(u) - J(\tilde{u}) = \frac{1}{2}\rho(\tilde{u})(z - \tilde{z}) + \frac{1}{2}\rho^*(\tilde{u}, \tilde{z})(u - \tilde{u}) + \rho(\tilde{u})(\tilde{z}) + \mathcal{R}^{(3)}, \tag{190}$$

where

$$\rho(\tilde{u})(\cdot) := -\mathcal{A}(\tilde{u})(\cdot), \tag{191}$$

$$\rho^*(\tilde{u}, \tilde{z})(\cdot) := J'(u) - \mathcal{A}'(\tilde{u})(\cdot, \tilde{z}), \tag{192}$$

and the remainder term

$$\mathcal{R}^{(3)} := \frac{1}{2}\int_0^1 [J'''(\tilde{u} + se)(e, e, e) \tag{193}$$

$$- \mathcal{A}'''(\tilde{u} + se)(e, e, e, \tilde{z} + se^*) - 3\mathcal{A}''(\tilde{u} + se)(e, e, e^*)]s(s - 1)\, ds, \tag{194}$$

with $e = u - \tilde{u}$ and $e^* = z - \tilde{z}$.

Proof. For the completeness of the presentation we add the proof below, which is very similar to [107]. First we define $x := (u, z) \in X := U \times V$ and $\tilde{x} := (\tilde{u}, \tilde{z}) \in X$. By assuming that $\mathcal{A} \in \mathcal{C}^3(U, V)$ and $J \in \mathcal{C}^3(U, \mathbb{R})$ we know that the Lagrange function

$$\mathcal{L}(\hat{x}) := J(\hat{u}) - \mathcal{A}(\hat{u})(\hat{z}) \quad \forall (\hat{u}, \hat{z}) =: \hat{x} \in X,$$

is in $\mathcal{C}^3(X, \mathbb{R})$. Assuming this, it holds

$$\mathcal{L}(x) - \mathcal{L}(\tilde{x}) = \int_0^1 \mathcal{L}'(\tilde{x} + s(x - \tilde{x}))(x - \tilde{x})\, ds.$$

Using the trapezoidal rule [107], we obtain

$$\int_0^1 f(s)\, ds = \frac{1}{2}(f(0) + f(1)) + \frac{1}{2}\int_0^1 f''(s)s(s - 1)\, ds,$$

for $f(s) := \mathcal{L}'(\tilde{x} + s(x - \tilde{x}))(x - \tilde{x})$ we conclude

$$\mathcal{L}(x) - \mathcal{L}(\tilde{x}) = \frac{1}{2}(\mathcal{L}'(x)(x - \tilde{x}) + \mathcal{L}'(\tilde{x})(x - \tilde{x})) + \mathcal{R}^{(3)}.$$

From the definition of $\mathcal{L}$ we observe that

$$\begin{aligned} J(u) - J(\tilde{u}) &= \mathcal{L}(x) - \mathcal{L}(\tilde{x}) + \underbrace{A(u)(z)}_{=0} - A(\tilde{u})(\tilde{z}) \\ &= \frac{1}{2}(\mathcal{L}'(x)(x - \tilde{x}) + \mathcal{L}'(\tilde{x})(x - \tilde{x})) - A(\tilde{u})(\tilde{z}) + \mathcal{R}^{(3)}. \end{aligned}$$

It remains to show that $\frac{1}{2}(\mathcal{L}'(x)(x - \tilde{x}) + \mathcal{L}'(\tilde{x})(x - \tilde{x})) = \frac{1}{2}\rho(\tilde{u})(z - \tilde{z}) + \frac{1}{2}\rho^*(\tilde{u}, \tilde{z})(u - \tilde{u})$. But this is true since

$$\begin{aligned} &\mathcal{L}'(x)(x - \tilde{x}) + \mathcal{L}'(\tilde{x})(x - \tilde{x}) \\ &= \underbrace{J'(u)(e) - \mathcal{A}'(u)(e, z)}_{=0} - \underbrace{A(u)(e^*)}_{=0} + \underbrace{J'(\tilde{u})(e) - \mathcal{A}'(\tilde{u})(e, \tilde{z})}_{=\rho^*(\tilde{u},\tilde{z})(u-\tilde{u})} - \underbrace{A(\tilde{u})(e^*)}_{=-\rho(\tilde{u})(z-\tilde{z})}. \end{aligned}$$

□

9.14.2 Adaptive fixed-point

See [107][Section 3.2].

9.14.3 Adaptive Newton

See [107][Section 3.3].

9.15 Effectivity of goal-oriented error estimation

Since $J(\cdot)$ is only a functional, in general not a norm, we can easily have

$$|J(u) - J(u_h)| = 0$$

while $u \neq u_h$. The objective in this section is to show that the error estimator η and $J(u) - J(u_h)$ have a common upper bound. If this is the case, we call the error estimator **effective**; see [113]. To show effectivity of the classical error estimator, we refer the reader to [113][Section 4.1]. The PU localization is treated in the same paper in Section 4.3.

9.16 Efficiency estimates using a saturation assumption

An improvement of the previous section was made in [46]. The prize to pay was a saturation assumption that is required in the corresponding proofs for the efficiency.

9.17 Principles of multi-goal-oriented error estimation

In the previous section we recapitulated the DWR method for computing a single goal functional. Now we assume that we are given N linear functionals with $N \in \mathbb{N}$. Let $\mathbb{J}$ be defined as $\mathbb{J} := \{J_0, \ldots, J_{N-1}\}$. We can use (188) for each $J_i \in \mathbb{J}$ where $i \in \mathbb{N}$ to compute the node-wise contributions of the error. But to do so we have to solve N adjoint problems. Therefore we seek a method to avoid these computations. We follow the idea from [71, 72] of combining the functionals in $\mathbb{J}$. We create a linear combination of the goal functionals to one functional $\tilde{J}_c$ resulting in

$$\tilde{J}_c(\psi) := \sum_{i=0}^{N-1} w_i J_i(\psi) \quad \forall \psi \in V, \tag{195}$$

where $w_i \in \mathbb{R}$. We call $\tilde{J}_c$ the combined functional. Now we have to find out how to choose the weights w_i. One crucial aspect is the sign of w_i because it may lead to error canceling. Furthermore, we are interested in having similar relative errors in our functional evaluations. One idea (as in [71, 72]) is to choose w_i as

$$w_i := \frac{sign(J_i(u) - J_i(u_h))\omega_i}{|J(u_h)|}, \tag{196}$$

where ω_i describes some self-chosen, but positive weights. This choice leads to no error canceling and also the relative errors are similar (if the weights ω_i are almost equal). The previous choices are reasonable because in the final end we are interested in the error

$$\tilde{J}_c(u - u_h) = \sum_{i=0}^{N-1} \frac{sign(J_i(u - u_h))\omega_i}{|J(u_h)|} J_i(u - u_h)$$

Unfortunately we do not know $J_i(u)$. Hence, we have to find a way to get $sign(J_i(u) - J_i(u_h))$. To do so, we consider the adjoint to the adjoint problem (which is akin to saying a discrete error problem) [71, 72]:

Formulation 9.50 (Adjoint to the adjoint problem)**.** *Find the error function e such that*

$$a(e, \psi) = \langle R_{u_h}, \psi \rangle \quad \forall \psi \in V, \tag{197}$$

where $\langle R_{u_h}, \psi \rangle := \langle f, \psi \rangle - a(u_h, \psi)$.

By solving this problem, we obtain e where $e = u - u_h$ and therefore we can compute $J_i(u) - J_i(u_h)$. The adjoint to the adjoint problem provides information with respect to the error in the goal functionals $J_i(u)$, but it does not yield local error information, which are required for mesh refinement. Thus the solution is to solve both the adjoint to the adjoint problem and another adjoint problem leading to two additional problems. In summary, using this approach for multiple goal functionals, three problems need to be solved: primal, adjoint, adjoint to the adjoint.

For treating the adjoint to the adjoint problem we again have to solve a PDE discretized by finite elements. For this problem we have to use a discrete subspace $V_h^{(r+1)} \subset V$ which fulfills $V_h \subsetneq V_h^{(r+1)}$ because otherwise $\langle R_{u_h}, \psi_h \rangle = 0$ for all $\psi_h \in V_h$, which must be avoided. Moreover, we have to solve the primal problem to compute u_h and then compute e as solution of the adjoint to the adjoint problem, so we have to solve two systems sequentially.

The dependence of the adjoint and adjoint to the adjoint problem slightly limits the possibility to further reduce the computational cost. Therefore, we suggest the following alternative in case we approximate the solution in our discrete subspace $V_h^{(r+1)}$, $(r \geq 1)$:

Proposition 9.51. *Let $a : V \times V \to \mathbb{R}$ be a bilinear form, $f \in V^*$, where V^* is the dual space of V, fulfilling the assumptions of Lax-Milgram (e.g., [51, 65]) and let $u_h, u_h^{(2)}, e_h^{(2)}$ be the solutions of the problems, respectively: Find $u_h \in V_h, e_h^{(2)}, u_h^{(2)} \in V_h^{(2)}$ such that*

$$a(u_h, \psi_h) = \langle f, \psi_h \rangle \quad \forall \psi_h \in V_h, \tag{198}$$

$$a(u_h^{(2)}, \psi_h^{(2)}) = \langle f, \psi_h^{(2)} \rangle \quad \forall \psi_h^{(2)} \in V_h^{(2)}, \tag{199}$$

and

$$a(e_h^{(2)}, \psi_h^{(2)}) = \langle R_{u_h}, \psi_h^{(2)} \rangle \quad \forall \psi_h^{(2)} \in V_h^{(2)}, \tag{200}$$

where $V_h \subset V$, $V_h^{(2)} \subset V$ and $\langle R_{u_h}, \psi_h^{(2)} \rangle$ are defined as in Formulation 9.50. Then there exists a projection $P_{[V_h^{(2)}]} : V \to V_h^{(2)}$ such that

$$e_h^{(2)} = u_h^{(2)} - P_{[V_h^{(2)}]} u_h. \tag{201}$$

Specifically, if $V_h \subseteq V_h^{(2)}$, it holds

$$e_h^{(2)} = u_h^{(2)} - u_h. \tag{202}$$

Proof. Let u_h be the solution of (198), then there exists a unique $f_{u_h} \in V^*$ such that

$$a(u_h, \psi) = \langle f_{u_h}, \psi \rangle \quad \forall \psi \in V. \tag{203}$$

If we want to approximate the solution of (203) on the finite element space $V_h^{(2)}$, we obtain the approximation u_{u_h} of u_h which is given by the unique solution of the problem: Find $u_{u_h} \in V_h^{(2)}$ such that

$$a(u_{u_h}, \psi_h^{(2)}) = \langle f_{u_h}, \psi_h^{(2)} \rangle \quad \forall \psi_h^{(2)} \in V_h^{(2)}.$$

It can be shown that the mapping $u_h \mapsto u_{u_h}$ is a projection, which is denoted by $P_{[V_h^{(2)}]}$.

For this projection holds:

$$a(u_h - \underbrace{P_{[V_h^{(2)}]} u_h}_{u_{u_h}}, \psi_h^{(2)}) = \langle f_{u_h}, \psi_h^{(2)} \rangle - \langle f_{u_h}, \psi_h^{(2)} \rangle = 0 \quad \forall \psi_h^{(2)} \in V_h^{(2)}.$$

A simple calculation shows

$$\begin{aligned} a(e_h^{(2)}, \psi_h^{(2)}) &= \langle R_{u_h}, \psi_h^{(2)} \rangle = \langle f, \psi_h^{(2)} \rangle - a(u_h, \psi_h^{(2)}) \\ &= a(u_h^{(2)}, \psi_h^{(2)}) - a(P_{[V_h^{(2)}]} u_h, \psi_h^{(2)}) \\ &= a(u_h^{(2)} - P_{[V_h^{(2)}]} u_h, \psi_h^{(2)}) \quad \forall \psi_h^{(2)} \in V_h^{(2)}. \end{aligned}$$

The Lax-Milgram lemma yields a unique solution in the space $V_h^{(2)}$ and we can conclude that $e_h^{(2)} = u_h^{(2)} - P_{[V_h^{(2)}]} u_h$ hence

$$a(e_h^{(2)}, \psi_h^{(2)}) = \langle R_{u_h}, \psi_h^{(2)} \rangle = a(u_h^{(2)} - P_{[V_h^{(2)}]} u_h, \psi_h^{(2)}) \quad \forall \psi_h^{(2)} \in V_h^{(2)}.$$

And this shows the first statement (201). If $V_h \subseteq V_h^{(2)}$ holds, then $P_{[V_h^{(2)}]} u_h = u_h$ because $u_h \in V_h^{(2)}$ and henceforth (202) has been shown. □

Remark 9.52. *The assumptions of the Lax-Milgram lemma can be relaxed by any condition which guarantees only that $a(u, \psi) = \langle f, \psi \rangle$ for all $\psi \in V$, (198), (199) and (200) have unique solutions for all $f \in V^*$.*

Remark 9.53. *Furthermore, we notice that our previous theory does hold not only for $V_h^{(1)} \subset V_h^{(r+1)}$ but for general spaces which are not necessarily subspaces of $V_h^{(r+1)}$.*

Corollary 9.54. *From Proposition 9.51 we obtain that if we work on the spaces V_h and $V_h^{(r+1)}$ the error can be simply computed as:*

$$e_h^{(r+1)} = u_h^{(r+1)} - u_h.$$

In particular, the two subproblems for obtaining $u_h^{(r+1)}$ and u_h can be computed in parallel without communication using the spaces V_h and $V_h^{(r+1)}$.

Furthermore to avoid problems with prolongation operators in programming, we can compute immediately $J_i(u_h)$ and just communicate this value. With the help of Proposition 9.51, the combined functional $\tilde{J}_c$ is approximated by J_c with

$$J_c(\psi) := \sum_{i=0}^{N-1} J_i(\psi) \frac{sign(J_i(u_h^{(r+1)}) - J_i(u_h))\omega_i}{|J_i(u_h)|} \quad \forall \psi \in V, \tag{204}$$

for some self-chosen but positive weights $\omega_i \in \mathbb{R}$. Now we can use the PU approach for the functional J_c in a similar way as discussed in Section 9.12.3, resulting in

Proposition 9.55. *For the finite element approximation of Formulation 9.27, and considering N goal functionals $J_i(\cdot)$ we have the a posteriori error estimate:*

$$|J_c(u) - J_c(u_h)| \leq \eta := \sum_{i=1}^{M} |\eta_i|, \tag{205}$$

where

$$\eta_i = \Big\{ \langle f, (z - i_h z)\psi_i \rangle - (\alpha \nabla u_h, \nabla (z - i_h z)\psi_i) \Big\}. \tag{206}$$

Specifically, the adjoint problem is given by: Find $z \in V$ such that

$$a(\psi, z) = J_c(\psi) \quad \forall \psi \in V,$$

for which J_c has been constructed by (204).

Remark 9.56. *Since in praxis we are in general not able to compute the exact adjoint solution z (see also Section 9.9.1), we approximate z by z_h solving the problem: Find $z_h \in V_h^{(r+1)}$ such that*

$$a(\psi_h, z_h) = J_c(\psi_h) \quad \forall \psi_h \in V_h^{(r+1)},$$

for which J_c has been constructed by (204).

Remark 9.57. *An advantage of this approach is that we have to solve (as in [71, 72]) only 2 linear systems instead of N, and furthermore we do not lose the possibility of parallelization.*

Remark 9.58. *If we use the same finite element space for the second primal problem and the adjoint problem, then we just have to assemble one matrix instead of the two system matrices $A_{primal}, A_{adjoint}$ since it holds $A_{adjoint} = A_{primal}^T$.*

9.17.1 The adaptive algorithm

Let error tolerances TOL_i be given for each $J_i \in \mathbb{J}$ and $i \in \{N\}$ where N is the number of functionals of interest. Mesh adaptation is realized by extracting local error indicators η_i from the a posteriori error estimate in Proposition 9.55 in which the adjoint solution has been approximated as explained in Remark 9.56. To this end, we can adapt the mesh using the following strategy:

1. **Solve two primal problems:** Compute the primal solutions u_h and $u_h^{(r+1)}$ for two finite element spaces, respectively. This can be done completely in parallel.
2. **Construct combined functional:** Construct J_c as in (204).
3. **Solve the adjoint problem:** Compute the adjoint solution $z_h^{(r+1)}$ by solving the adjoint problem $a(\psi_h, z_h) = J_c(\psi_h)$ on a larger FE-space than for u_h.
4. **Estimate:**
   - Determine the indicator η_i at each node i by (206).
   - Compute the sum of all indicators $\eta := \sum_i \eta_i$.
   - Check, if the stopping criterion is satisfied:

     $$\eta < TOL_c,$$

     where $TOL_c := \inf_{i \in \{N\}} \{ \frac{\omega_i TOL_i}{|J_i(u_h)|} \}$. If this criterion is satisfied, stop the computation since $J_c(u_h)$ has been computed with a desired accuracy. Otherwise, proceed to the following step.
5. **Mark** all cells K_i that touch nodes that have values η_i above the average $\frac{\alpha \eta}{N}$ (where N denotes the total number of cells of the mesh $\mathbb{T}_h$ and $\alpha \approx 1$).

6. **Refine** the mesh.

7. **Go back to 1.**

Remark 9.59. *The reason for the special choice of TOL_c is to ensure that for all functionals J_i holds that $|J_i(u) - J_i(u_h)| < TOL_i$ in which we tacitly assume that the exact adjoint solution would be known.*

Remark 9.60. *Despite that we formulated our algorithm for r and $r+1$, we notice that we could also have worked with the same polynomial degree but locally using finer meshes to obtain the second space. This is very similar to the various options that we have to approximate the adjoint problem itself as described in Section 9.9.1.*

9.18 Residual-based error estimation

Setting

$$J(\phi) = \|\nabla e_h\|^{-1}(\nabla\phi, \nabla e_h)$$

we obtain an estimator for the energy norm. And setting

$$J(\phi) = \|e_h\|^{-1}(\phi, e_h)$$

we obtain an estimator for the L^2 norm. For details we refer to [15] and [106].

9.19 Mesh refinement strategies: averaging, fixed-rate, fixed-fraction

We have now on each element $K_j \in \mathcal{T}_h$ or each PU-DoF i an error value. It remains to set-up a strategy which elements shall be refined to enhance the accuracy in terms of the goal functional $J(\cdot)$.

Let an error tolerance TOL be given. Mesh adaption is realized using extracted local error indicators from the a posteriori error estimate on the mesh T_h. A cell-wise assembling reads:

$$|J(u) - J(u_h)| \leq \eta := \sum_{K \in T_h} \eta_K \quad \text{for all cells } K \in T_h.$$

Alternatively, the PU allows for a DoF-wise assembling:

$$|J(u) - J(u_h)| \leq \eta := \sum_{i} \eta_i \quad \text{for all DoFs } i \text{ of the PU.}$$

9.19.1 Averaging and general algorithm

This information is used to adapt the mesh using the following strategy:

1. Compute the primal solution u_h and the adjoint solution z_h on the present mesh T_h.

2. Determine the cell indicator η_K at each cell K.
   Alternatively, determine the DoF-indicator η_i at each PU-DoF i.

3. Compute the sum of all indicators $\eta := \sum_{K \in T_h} \eta_K$.
   Alternatively, $\eta := \sum_i \eta_i$.

4. Check, if the stopping criterion is satisfied: $|J(u) - J(u_h)| \leq \eta \leq TOL$, then accept u_h within the tolerance TOL. Otherwise, proceed to the following step.

5. Mark all cells K_i that have values η_{K_i} above the average $\frac{\alpha\eta}{N}$ (where N denotes the total number of cells of the mesh $\mathbb{T}_h$ and $\alpha \approx 1$).
   Alternatively, all PU-DoFs are marked that are above, say, the average $\frac{\alpha\eta}{N}$.

Remark 9.61. *We emphasize that the tolerance TOL should be well above the tolerances of the numerical solvers. Just recall $TOL = 0.01$ would mean that the goal functional is measured up to a tolerance of 1%.*

When the DoF-based estimator is adopted, the error indicators η_i are node-wise contributions of the error. Mesh adaptivity can be carried out in two ways:

- in a node-wise fashion: if a node i is picked for refinement, all elements touching this node will be refined;
- alternatively, one could also first assemble element wise for each $K \in \mathcal{T}_h$ indicators by summing up all indicators belonging to nodes of this element and then carry out adaptivity in the usual element-wise way.

On adaptive meshes with hanging nodes, the evaluation of the PU indicator is straightforward: First, the PU is assembled in (189) employing the basis functions $\psi_i \in V_{PU}$ for $i = 1, \dots, M$. In a second step, the contributions belonging to hanging nodes are condensed in the usual way by distribution to the neighboring indicators.

9.19.2 Fixed-rate (fixed number)

See for instance [11, 15]. Refining/coarsening a fixed fraction/number of elements. All elements are ordered with respect to their error values:

$$\eta_1 \geq \eta_2 \geq ... \geq \eta_N.$$

Let X and Y be fractions with $1 - X > Y$. Here, $X \cdot N$ elements are refined and $Y \cdot N$ elementss are coarsened. For instance, $X = 0.3$ and $Y = 0.02$.

9.19.3 Fixed error reduction (fixed-fraction)

Refining/coarsening according to a reduction of the error estimate (also known as bulk criterion or Dörfler marking [42]). Here, the error values are summed up such that a prescribed fraction of the total error is reduced. All elements that contribute to this fraction are refined. Let again X and Y be fractions with $1 - X > Y$. Determine indices $N_l, N_u \in \{1, \dots, N\}$ with N being the actual size of the system, such that

$$\sum_{i=1}^{N_u} \eta_i \approx X\eta, \quad \sum_{i=N_l}^{N} \eta_i \approx Y\eta.$$

Then, the elements $K_1, \dots, K_{N_u}$ are refined and $K_{N_l}, \dots, K_N$ are coarsened. For instance, $X = 0.2$ and $Y = 0.1$.

9.19.4 How to refine marked cells

It remains to explain how marked cells are finally refined.

9.19.4.1 Quads and hexs Using quadrilateral or hexahedra meshes, simply bisection can be used. Here, a cell is cut in the middle and split into 4 (in 2d) or 8 (in 3d) sub-elements. When the neighboring cell is not refined, we end up with so-called **hanging nodes**. These are degrees of freedom on the refined cell, but on the coarse neighboring cell, these nodes lie on the middle point of faces or edges and do not represent true degrees of freedom. Their values are obtained by interpolation of the neighboring DoFs. Consequently, condition No. 3 in Definition 8.127 is weakened in the presence of hanging nodes. For more details, we refer to Carey and Oden [30].

9.19.4.2 Triangles and prims For triangles or prisms, we have various possibilies how to split elements into sub-elements. Here, common ways are red (one triangle is split into four triangles) and green (one triangle is split into two triangles) refinement strategies. Here a strategy is to use red refinement and apply green refinement when hanging nodes would occur in neighboring elements.

9.19.4.3 Conditions on the geometry While refining the mesh locally for problems in $n \geq 2$, we need to take care that the minimum angle condition (see Section 8.14) is fulfilled.

9.19.5 Convergence of adaptive algorithms

The first convergence result of an adaptive algorithm was shown in [42] for the Poisson problem. The convergence of adaptive algorithms of generalized problems is subject to current research. Axioms of adaptivity have been recently formulated in [31].

9.20 Numerical test: Poisson with mean value goal functional

We consider the following test first: Let $\Omega = (0,1)^2$ and find $u \in H_0^1(\Omega)$ such that

$$(\nabla u, \nabla \phi) = (1, \phi) \quad \forall \phi \in H_0^1.$$

The goal functional is

$$J(u) = \frac{1}{|\Omega|} \int_\Omega u \, dx.$$

The reference value has been computed on a very fine mesh (despite here a manufactured solution would have been easy to derive) and is given by

$$J_{ref}(u) = 3.5144241236e-02.$$

The above goal functional yields the adjoint problem:

$$a(\phi, z) = J(\phi)$$

and in particular:

$$(\nabla \phi, \nabla z) = \int_\Omega \phi \, dx = (1, \phi).$$

The parameter α in Section 9.19 is chosen $\alpha = 4$. The solution of the adjoint problem is the same as the primal problem as also illustrated in Figure 33.

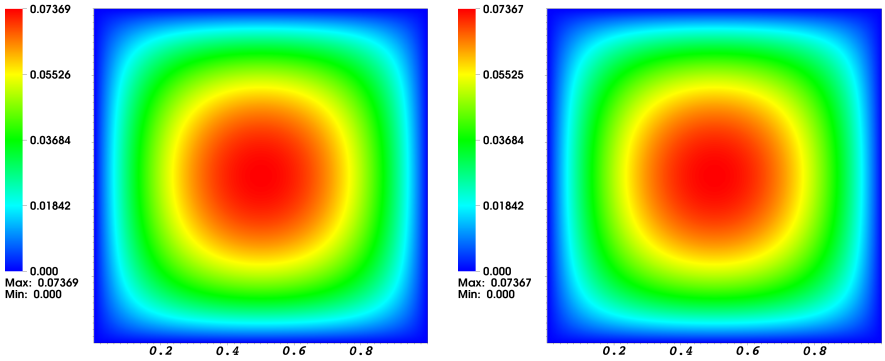


Figure 33: Section 9.20: Solutions of the primal and adjoint problems.

The first goal in our computations is to investigate I_{eff} using the PU localization from Proposition 9.47; again in short form:

$$|J(u) - J(u_h)| \leq \eta := \sum_{i=1}^{M} \left| \langle f, (z_h^{(2)} - i_h z_h^{(2)}) \psi_i \rangle - (\nabla u_h, \nabla (z_h^{(2)} - i_h z_h^{(2)}) \psi_i) \right|, \tag{207}$$

A second goal is to show that a pure weak form without partial integration or without PU does yield an over-estimation of the error:

$$|J(u) - J(u_h)| \leq \eta := \sum_{i=1}^{M} \left| \langle f, (z_h^{(2)} - i_h z_h^{(2)}) \rangle - (\nabla u_h, \nabla (z_h^{(2)} - i_h z_h^{(2)})) \right|, \tag{208}$$

Findings using PU localization

Dofs	True error	Estimated error	Effectivity
25	3.17e-03	3.14e-03	9.92e-01
69	1.66e-03	2.23e-03	1.35e+00
153	7.15e-04	1.02e-03	1.43e+00
329	2.01e-04	2.19e-04	1.09e+00
829	1.07e-04	1.43e-04	1.34e+00
1545	4.83e-05	5.62e-05	1.16e+00
4049	1.68e-05	2.28e-05	1.35e+00
6673	1.18e-05	1.35e-05	1.15e+00
15893	4.15e-06	5.35e-06	1.29e+00
24853	2.96e-06	3.29e-06	1.11e+00
62325	1.05e-06	1.38e-06	1.31e+00
96953	7.34e-07	8.21e-07	1.12e+00

Findings using a pure weak form without part. int. nor PU

Dofs	True error	Estimated error	Effectivity
25	3.17e-03	1.26e-02	3.97e+00
69	1.66e-03	8.79e-03	5.31e+00
165	6.92e-04	4.12e-03	5.95e+00
313	3.47e-04	2.42e-03	6.98e+00
629	1.77e-04	1.02e-03	5.73e+00
1397	8.36e-05	5.64e-04	6.75e+00
2277	4.57e-05	2.37e-04	5.19e+00
4949	2.14e-05	1.18e-04	5.54e+00
8921	1.04e-05	5.10e-05	4.90e+00
22469	4.71e-06	2.73e-05	5.78e+00
37377	2.66e-06	1.36e-05	5.12e+00
80129	1.83e-06	8.85e-06	4.83e+00

In the second table, we clearly observe an over-estimation of the error by a factor of 5. In the case of the PU localization, we see I_{eff} around 1, what we expected due to the regularity of the problem.

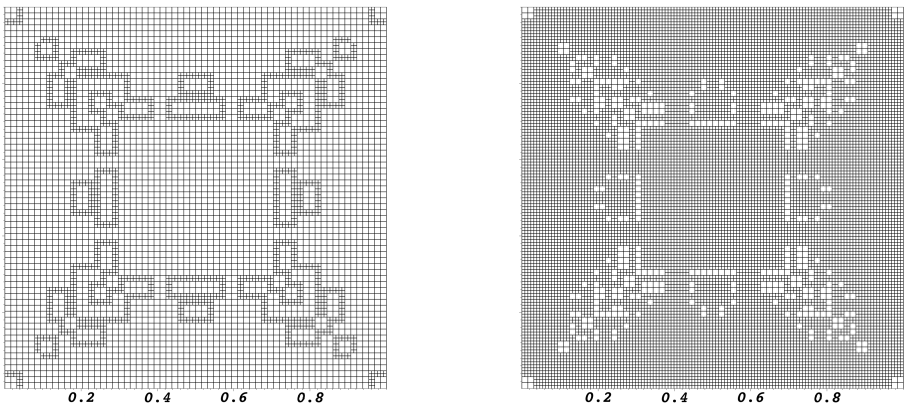


Figure 34: Section 9.20: Locally adapted meshes using PU localization for the refinement levels 7 and 8.

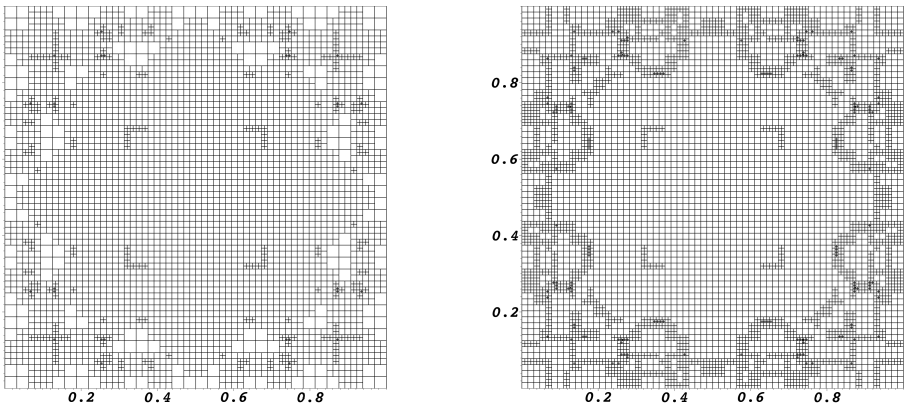


Figure 35: Section 9.20: Locally adapted meshes using a pure weak form without partial integration nor PU localization for the refinement levels 7 and 8.

9.21 Numerical test: L-shaped domain with Dirac rhs and Dirac goal functional

We redo Example 2 from [113]. Specifically, we consider Poisson's equation on an L-shaped domain $\Omega_L = (-1,1)^2 \setminus (-1,0)^2$ with bounddary $\partial\Omega_L$, where the right hand side is given by a Dirac function in $x_0 = (-0.5, 0.5)$

$$-\Delta u = \delta_{x_0} \text{ in } \Omega_L, \quad u = 0 \text{ on } \partial\Omega_L. \tag{209}$$

As goal functional, we take the point evaluation in $x_1 = (0.5, -0.5)$ such that $J(\phi) = \phi(x_1)$. The adjoint problem corresponds to solving Poisson's equation $-\Delta z = \delta_{x_1}$ with a Dirac right hand side in x_1. Both the primal problem and the adjoint problem lack the required minimal regularity for the standard finite element theory (see the previous chapter), such that a regularization by averaging is required, e.g. by averaging over a small subdomain:

$$J_\epsilon(\phi) = \frac{1}{2\pi\epsilon^2} \int_{|x-x_1|<\epsilon} \phi(x)\,\mathrm{d}x, \tag{210}$$

where $\epsilon > 0$ is a small parameter not depending on h. As reference functional quantity we compute the value

$$J_{ref} = 2.134929 \cdot 10^{-3} \pm 10^{-7} \tag{211}$$

obtained on a sufficiently globally refined mesh.

Dofs	True error	Estimated error	Effectivity
21	1.24e-03	1.41e-03	1.14e+00
65	2.74e-04	2.45e-04	8.97e-01
225	7.50e-05	6.32e-05	8.42e-01
677	2.65e-05	2.11e-05	7.95e-01
2021	9.93e-06	7.70e-06	7.75e-01
6089	3.59e-06	2.80e-06	7.81e-01
20157	1.38e-06	1.18e-06	8.51e-01
61905	5.16e-07	5.20e-07	1.01e+00
177221	1.34e-07	2.10e-07	1.56e+00

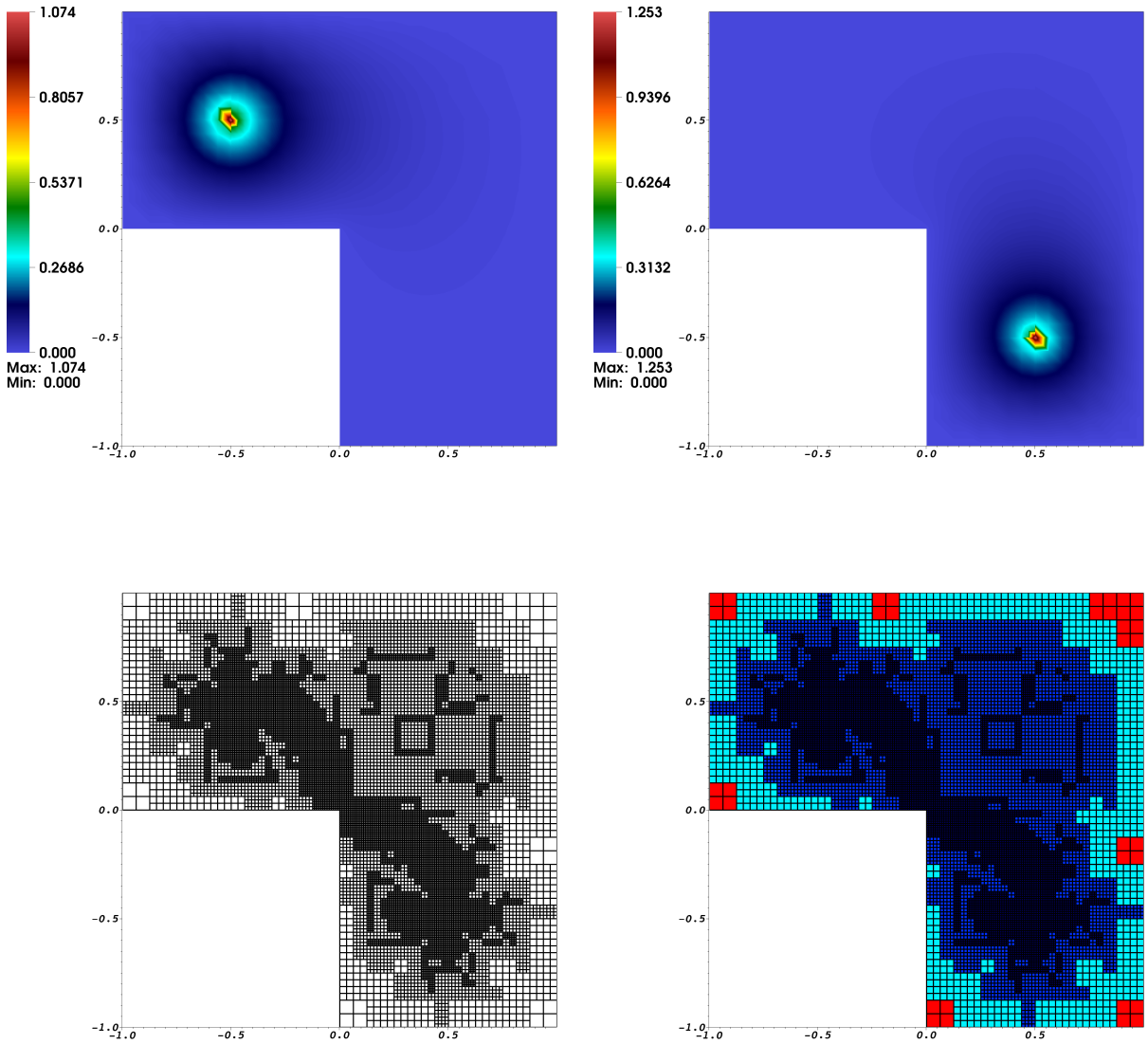


Figure 36: Section 9.21: Poisson problem with Dirac right hand side (left figure on top) and Dirac goal functional (right figure on top) on the l-shaped domain. For mesh refinement the primal error estimator with PU localization is used. The primal problem is discretized with Q_1 elements and the adjoint problem with Q_2 elements. The PU is based on Q_1 elements. The mesh is refined around the two Dirac functions and in the corner of the domain (bottom figure left). In order to highlight the various mesh levels within one computation, the different element areas are colorized (bottom figure right).

9.22 Space-time goal-oriented error estimation

See [11][Section 3 and Section 9] and [19].

9.23 Final comments to error estimation in numerical simulations

We give some final comments to error estimation. One should keep in mind several aspects. First:

- Numerical mathematics is its own discipline and we should aim to develop methods with best accuracy as possible.
- On the other hand, numerical mathematics is a very useful tool for many other disciplines (engineering, natural sciences, etc.). Here, an improvement of a process (i.e., approximation) by several percent is already very important, while the total error may be still large though and in the very sense of numerical mathematics the result is not yet satisfying.

Second (w.r.t. the first bullet point): A posteriori error estimation, verification of the developed methods, and **code validation** are achieved in **three steps** (with increasing level of difficulty) after having constructed an a posteriori error estimator η that can be localized and used for local mesh adaptivity. These steps hold true for classical norm-based error-estimation $\|u - u_h\|$ or goal-oriented error estimation $J(u) - J(u_h)$.

1. Does the solution make sense?
   - If possible test your code with an acknowledged benchmark configuration and verify whether $J(u_h)$ matches the benchmark values in a given range and on a sequence of at least **three meshes**. This first step can be performed with uniform and adaptive mesh refinement. In time-dependent problems, please compute at least with three different time step sizes.
   - If no benchmark configuration available, study a **simple, prototype** configuration and observe whether the solution makes sense.
2. Do the true error $J(u) - J(u_h)$ and the error estimator η descrease their values under mesh refinement?
   - Compute $J(u)$ either on a uniformly-refined super-fine mesh or even analytically (i.e., a manufactured solution). Compute the error $J(u) - J(u_h)$ and observe whether the error is decreasing.
   - If a priori estimates are available, see if the orders of convergence are as expected. But be careful, often goal functionals are nonlinear, for which rigorous a priori error estimates are often not available.
3. Compare η and $J(u)-J(u_h)$ in terms of the effectivity index I_{eff}. Do we asymptotically obtain something around $I_{eff} \approx 1$? For nonlinear problems, one easily observes $0.5 \leq I_{eff} \leq 10$, which might be still okay depending on the problem.

In the very sense of numerical mathematics we must go for all three steps, but in particular the last step No. 3 is often very difficult when not all theoretical requirements (smoothness of data and boundary conditions, regularity of the goal functional, smoothness of domains and boundaries) are fulfilled. Also keep in mind that all parts of error estimators are implemented and that still the remainder term R may play a role. For instance, in practice, very often when working with the DWR method, only the primal error estimator is implemented and the adjoint part neglected; see the following section.

9.24 Example: Stationary Navier-Stokes 2D-1 benchmark

We demonstrate the developements in this chapter with a well-known benchmark in fluid mechanics: the 2D-1 benchmark [118]. The underlying equations are the stationary incompressible Navier-Stokes equations [57, 102, 126], which are vector-valued (more details in Section 11) and nonlinear (more details in Section 13).

The goals are the accurate measurements of drag, lift and a pressure difference.

9.24.1 Equations

Formulation 9.62. *Let $\Omega \subset \mathbb{R}^2$ with boundary $\partial\Omega = \Gamma_{wall} \cup \Gamma_{in} \cup \Gamma_{out}$. Find vector-valued velocities $v : \Omega \to \mathbb{R}^2$ and a scalar-valued pressure $p : \Omega \to \mathbb{R}$ such that*

$$\begin{aligned} \rho(v \cdot \nabla)v - div(\sigma) &= \rho f \quad in\ \Omega, \\ div(v) &= 0 \quad in\ \Omega, \\ v &= 0 \quad on\ \Gamma_{wall}, \\ v &= g \quad on\ \Gamma_{in}, \\ \nu\partial_n v - pn &= 0 \quad on\ \Gamma_{out}, \end{aligned}$$

where $f : \Omega \to \mathbb{R}^2$ is a force, $g : \Gamma \to \mathbb{R}^2$ a Dirichlet inflow profile, n the normal vector, ρ the density and

$$\sigma = -pI + \rho\nu(\nabla v + \nabla v^T), \quad [\sigma] = Pa = N/m^2$$

is the Cauchy stress tensor for a Newtonian fluid. Moreover, I is the identity matrix and ν the kinematic viscosity.

The Reynolds number Re is given by the dimensionless expression

$$Re = \frac{LU}{\nu}$$

where L is a characteristic length and U a characteristic velocity.

Remark 9.63 (Stokes)**.** *Neglecting the convection term $(v \cdot \nabla)v$ for viscous flow, we obtain the linear Stokes equations.*

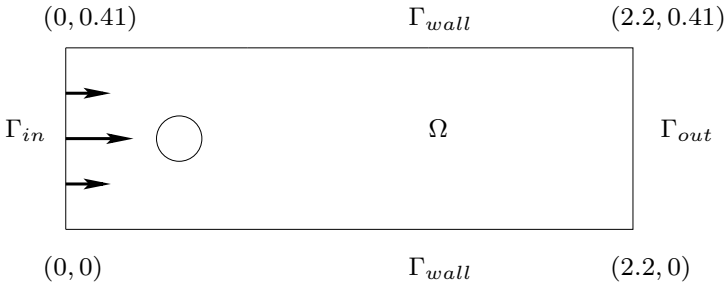


Figure 37: 2D-1 benchmark configuration.

For a variational formulation, we define the spaces (for details see [57, 102, 126]):

$$\begin{aligned} V_v &:= \{v \in H^1(\Omega)^2 |\ v = 0 \text{ on } \Gamma_{in} \cup \Gamma_{wall}\}, \\ V_p &= L^2(\Omega)/\mathbb{R}. \end{aligned}$$

Here, V_v is a vector-valued space. The pressure is only defined up to a constant. For existence of the pressure, the inf-sup condition is adopted [57, 102, 126].

On the outflow boundary Γ_{out}, we apply a natural boundary condition:

$$\rho\nu\partial_n v - pn = 0.$$

This type of condition implicitly normalizes the pressure [75, 101] to a unique solution. Thus, the pressure space can be written as

$$V_p = L^2(\Omega).$$

Remark 9.64. *Applying $\rho\nu\partial_n v - pn = 0$ on Γ_{out} is not consistent with the total stress $\sigma \cdot n = -pI + \rho\nu(\nabla v + \nabla v^T) \cdot n$. The symmetric part $\rho\nu\nabla v^T \cdot n$ needs to be subtracted on Γ_{out}. For more details see [75, 138]. For a better concentration on the main parts in this section, we neglect this term in the following.*

As variational formulation, we obtain:

Formulation 9.65. *Find $v \in \{g + V_v\}$ and $p \in V_p$ such that*

$$\begin{aligned}(\rho(v \cdot \nabla)v, \phi) + (\sigma, \nabla\phi) = (\rho f, \phi) \quad \forall \phi \in V_v,\\ (div(v), \psi) = 0 \quad \forall \psi \in V_p.\end{aligned}$$

For the prescription of non-homogeneous Dirichlet conditions, we refer to Section 8.11.3.

For the finite element spaces we have to satisfy a discrete inf-sup condition [57]. Basically speaking, the finite element space for the velocities must be large enough. A stable FE pair is the so-called Taylor-Hood element $Q_2 \times Q_1$; here defined on quadrilaterals. The velocities are bi-quadratic and the pressure bi-linear. Hence we seek $(v_h, p_h) \in V_h^2 \times V_h^1$ with

$$V_h^s := \{u_h \in C(\bar{\Omega}) |\ u_h|_K \in Q_s(K), \forall K \in \mathcal{T}_h\}.$$

This is a conforming finite element space, i.e., $V_h^s \subset H^1(\Omega)$.

Formulation 9.66. *Find $v_h \in \{g + V_h^2\}$ and $p_h \in V_h^1$ such that*

$$\begin{aligned}(\rho(v_h \cdot \nabla)v_h, \phi_h) + (\sigma, \nabla\phi_h) = (\rho f, \phi_h) \quad \forall \phi_h \in V_h^2,\\ (div(v_h), \psi_h) = 0 \quad \forall \psi_h \in V_h^1.\end{aligned}$$

A compact semi-linear form is formulated as follows:

Formulation 9.67. *Find $U_h := (v_h, p_h) \in \{g + V_h^2\} \times V_h^1$ such that*

$$a(U_h)(\Psi_h) = (\rho(v_h \cdot \nabla)v_h, \phi_h) + (\sigma, \nabla\phi_h) - (\rho f, \phi_h) + (div(v_h), \psi_h) \quad \forall \Psi_h = (\phi_h, \psi_h) \in V_h^2 \times V_h^1$$

9.24.2 Functionals of interest

The goal functionals are drag, lift, and pressure difference, respectively:

$$J_1(U) = \int_S \sigma \cdot n e_1 \, ds, \quad J_2(U) = \int_S \sigma \cdot n e_2 \, ds, \quad J_3(U) = p(x_0, y_0) - p(x_1, y_1)$$

with $x_0, x_1, y_0, y_1 \in \Omega$ and where e_1 and e_2 are the unit vectors in x and y direction. We are interested in the drag and lift coefficients:

$$c_D = \frac{2J_1}{\rho V^2 D}, \quad c_L = \frac{2J_2}{\rho V^2 D}$$

with $D = 0.1m$ (the cylinder diameter) and the mean velocity

$$V = V(t) = \frac{2}{3} v_x(0, H/2)$$

where v_x is defined below and with the height of the channel $H = 0.41m$.

9.24.3 A duality-based a posteriori error estimator (primal part)

We develop an error estimator only based on the primal part. Since the problem is nonlinear, the adjoint part should be better evaluated as well. On the other, the adjoint part is expensive for which reason it might be justified only to work with the primal error estimator.

The adjoint bi-linear form is given by:

Formulation 9.68. *From the optimality system, we obtain:*

$$a'_U(U_h)(\Psi_h, Z_h) = (\rho(\phi_h \cdot \nabla)v_h + \rho(v_h \cdot \nabla)\phi_h, z_h^v) + (-\psi_h I + \nu\rho(\nabla\phi_h + \nabla\phi_h^T), \nabla z_h^v) + (div(\phi_h), z_h^p).$$

Proposition 9.69 (Adjoint problem)**.** *The adjoint problem depends on the specific goal functional $J_i(U), i = 1, 2, 3$. For example, for the drag, it holds: Find $Z_h = (z_h^v, z_h^p) \in V_h^3 \times V_h^2$ such that*

$$a'_U(U_h)(\Psi_h, Z) = J_1(\Psi_h) \quad \forall \Psi_h = (\phi_h, \psi_h) \in V_h^3 \times V_h^2$$

with $a'_U(U_h)(\Psi_h, Z)$ defined in Formulation 9.68 and

$$J_1(\Psi_h) = \int_S \Big(-\psi_h I + \rho\nu(\nabla\phi_h + \nabla\phi_h^T)\Big) \cdot ne_1 \, ds, \quad \Psi_h = (\phi_h, \psi_h).$$

Recall that the adjoint solution must be approximated with a higher-order method. Therefore, we choose the space $V^3 \times V^2$.

The primal PU-DWR error estimator reads:

Proposition 9.70. *It holds:*

$$|J(U) - J(U_h)| \leq \eta := \sum_i |\eta_i|,$$

with

$$\eta_i = -a(U_h)((Z_h - i_h Z_h)\chi_i), \quad Z_h = (z_h^v, z_h^p), \quad \chi_i \in V_{PU}$$

and the PU is realized as $V_{PU} := V_h^1$ and where $a(U_h)(\cdot)$ has been defined in Formulation 9.67.

Remark 9.71. *In the error estimator the non-homogeneous Dirichlet data g on Γ_{in} are neglected and assumed to be small. Also, the outflow correction condition, see Remark 9.64, is neglected.*

9.24.4 2D-1 configuration

The configuration is displayed in [118][Figure 1].

9.24.5 Boundary conditions

As boundary conditions, we specified all boundaries previously except the inflow:

$$g = (g_1, g_2)^T = (v_x(0, y), v_y)^T = (4v_m y\frac{H - y}{H^2}, 0)^T$$

with $v_m = 0.3m/s$. The resulting Reynolds number is $Re = 20$ (use D and V as defined above).

9.24.6 Parameters and right hand side data

We use $\rho = 1kg/m^3, \nu = 10^{-3}m^2/s$ and $f = 0$.

9.24.7 Step 1: Verification of benchmark values

In Table 3 in [118], the following bounds are given:

- Drag, c_D: $5.57 - 5.59$.
- Lift, c_L: $0.0104 - 0.0110$.
- Pressure difference, Δp: $0.1172 - 0.1176$.

We obtain on level 3 with 1828 elements and with 15868 velocity DoFs and 2044 pressure DoFs:

```
Functional     Value
====================================
Drag:          5.5754236905876873e+00
Lift:          1.0970590684824442e-02
Pressure diff: 1.1745258793930979e-01
====================================
```

9.24.8 Step 2 and Step 3: Computing $J(u)$ on a fine mesh, $J(u) - J(u_h)$ and I_{eff}

The reference value $J_1(U)$ for the drag is:

```
5.5787294556197073e+00
```

and was obtained on a four times (Level 4) uniformely refined mesh.

We obtain:

```
Level   Exact err       Est err           I_eff
================================================
0       3.51e-01        1.08e-01          3.07e-01
1       9.25e-02        2.08e-02          2.25e-01
2       1.94e-02        6.07e-03          3.12e-01
3       3.31e-03        2.45e-03          7.40e-01
================================================
```

We observe that the true error and the estimated error η both decrease. The effectivity indicies are less than 1 and indicate an underestimation of the true error (clearly seen in columns No. 2 and 3 as well). Because of our several assumptions and relatively coarse meshes, the results are more or less satisfying w.r.t. Step 3. Regarding Step 1 and 2, our findings are excellent.

9.24.9 More findings - graphical solutions

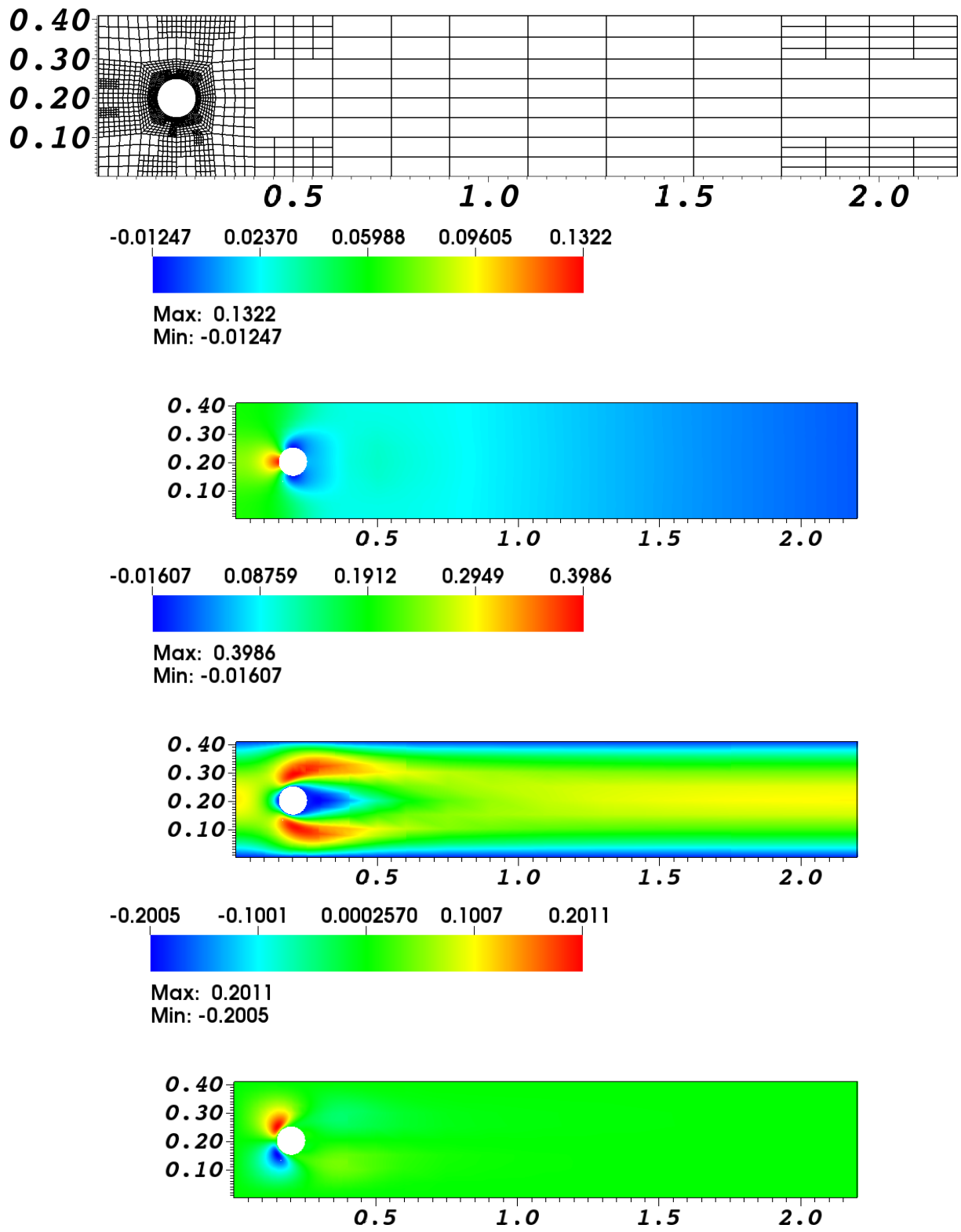


Figure 38: 2D-1 fluid benchmark: adaptively refined mesh with respect to $J_1(U)$, the pressure solution, x-velocity, y-velocity.

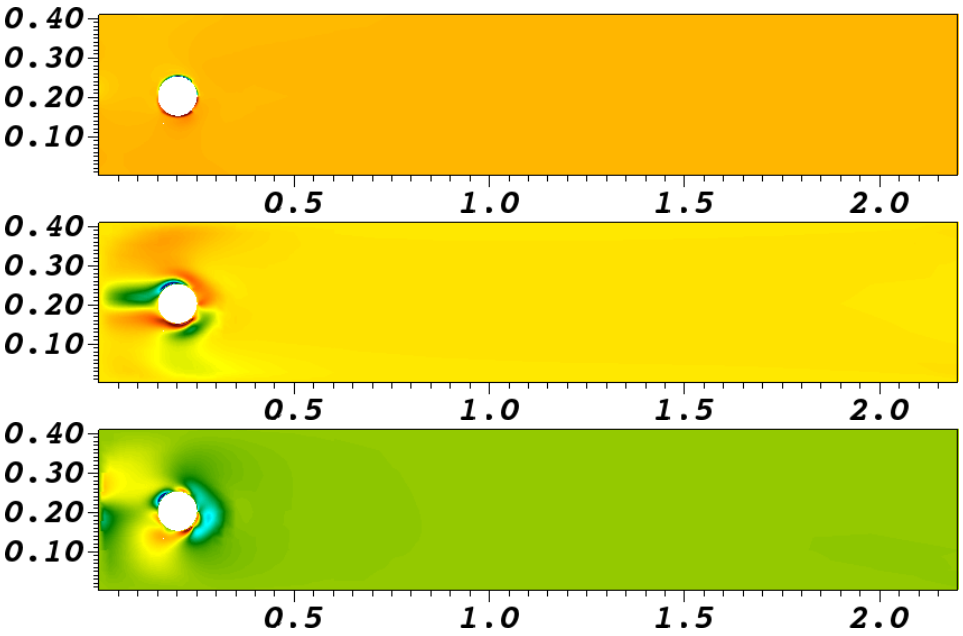


Figure 39: 2D-1 fluid benchmark: adjoint solutions of the pressure solution, x-velocity, y-velocity.

9.25 Example: Stationary fluid-structure interaction

The previous example can be extended to fluid-structure interaction. Details can be found:

- online on github https://github.com/tommeswick/goal-oriented-fsi
- a preprint https://arxiv.org/abs/2105.11145 for details on the problem statement and algorithms

9.26 Chapter summary and outlook

In this chapter, we introduced goal-oriented a posteriori error estimation to control the error during a finite element simulation a posteriori. Furthermore, these estimates can be localized on each mesh element and then be used for mesh refinement to obtain a higher accuracy of the numerical solution with low computational cost. The discrete systems in the present and the previous chapter lead to large linear equation systems that can in general not be solved anymore with a direct decomposition (e.g., LU). We shall briefly introduce some iterative methods in the next chapter.

10 Numerical solution of the discretized problems

In this chapter, we provide some ideas how to solve the arising linear systems

$$Ax = b$$

where

$$A \in \mathbb{R}^{n\times n}, \quad x = (u_1, \ldots, u_n)^T \in \mathbb{R}^n, \quad b \in \mathbb{R}^n.$$

when discretizing a PDE using finite differences or finite elements. We notice that to be consistent with the previous notation, we assume that the boundary points x_0 and x_{n+1} are not assembled.

For a moderate number of degrees of freedom, direct solvers such as Gaussian elimination, LU or Cholesky (for symmetric A) can be used.

More efficient schemes for large problems in terms of

- computational cost (CPU run time);
- and memory consumptions

are **iterative solvers.** Illustrative examples of floating point operations and CPU times are provided in [114][Pages 68-69, Tables 3.1 and 3.2].

We notice that most of the material in this chapter is taken from [114][Chapter 7] and translated from German into English.

10.1 On the condition number of the system matrix

Proposition 10.1. *Let $\mathcal{T}_h$ be a quasi-uniform triangulation. The condition numbers $\chi(A)$ of the system matrix A, and the condition number $\chi(M)$ of a mass matrix M can be estimated by:*

$$\chi(A) = O(h^{-2}), \qquad \chi(M) = O(1), \quad h \to 0.$$

Here

$$A = a_{ij} = a(\phi_j, \phi_i) = \int_\Omega \nabla\phi_j \nabla\phi_i \, dx,$$
$$M = m_{ij} = m(\phi_j, \phi_i) = \int_\Omega \phi_j \phi_i \, dx.$$

Proof. See [106], Section 3.5 or also [82], Section 7.3. □

10.2 Fixed-point schemes: Richardson, Jacobi, Gauss-Seidel

A large class of schemes is based on so-called **fixed point** methods:

$$g(x) = x.$$

We provide in the following a brief introduction that is based on [114]. Starting from

$$Ax = b$$

we write

$$0 = b - Ax$$

and therefore

$$x = \underbrace{x + (b - Ax)}_{g(x)}.$$

Introducing a scaling matrix C (in fact C is a preconditioner) and an iteration, we arrive at

$$x^k = x^{k-1} + C(b - Ax^{k-1}).$$

Summarizing, we have

Definition 10.2. *Let $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$ and $C \in \mathbb{R}^{n\times n}$. To solve*

$$Ax = b$$

we choose an initial guess $x^0 \in \mathbb{R}^n$ and we iterate for $k = 1, 2, \ldots$:

$$x^k = x^{k-1} + C(b - Ax^{k-1}).$$

Please be careful that k does not denote the power, but the current iteration index. Furthermore, we introduce:

$$B := I - CA \quad \textit{and} \quad c := Cb.$$

Then:

$$x^k = Bx^{k-1} + c.$$

Thanks to the construction of

$$g(x) = Bx + c = x + C(b - Ax)$$

it is trivial to see that in the limit $k \to \infty$, it holds

$$g(x) = x$$

with the solution

$$Ax = b$$

Remark 10.3. *Thanks to Banach's fixed point theorem (see again [114]), we can investigate under which conditions the above scheme will converge. We must ensure that*

$$\|g(x) - g(y)\| \leq \|B\| \, \|x - y\|$$

with $\|B\| < 1$. In fact we have:

$$g(x) = Bx + c, \quad g(y) = By + c$$

from which the above estimate is easily to see. A critical issue (which is also true for the ODE cases) is that different norms may predict different results. For instance it may happen that

$$\|B\|_2 < 1 \quad \textit{but} \quad \|B\|_\infty > 1.$$

For this reason, one often works with the spectral norm spr(B). More details can be found for instance in [114]. That we have a chance to achieve $\|B\| < 1$ is the reason why we introduced the matrix C.

We concentrate now on the algorithmic aspects. The two fundamental requirements for the matrix C (defined above) are:

- It should hold $C \approx A^{-1}$ and therefore $\|I - CA\| \ll 1$;
- It should be simple to construct C.

Of course, we easily see that these two requirements are conflicting statements. As always in numerics we need to find a trade-off that is satisfying for the developer and the computer.

10.2.1 On the Richardson iteration

We investigate the Richardson iteration in more detail.

Definition 10.4 (Richardson iteration)**.** *The simplest choice of C is the identity matrix, i.e.,*

$$C = \omega I.$$

Then, we obtain the Richardson iteration

$$x^k = x^{k-1} + \omega(b - Ax^{k-1})$$

with a relaxation parameter $\omega > 0$.

Let $B = I - \omega A$ be the iteration matrix of Richardson's method. The eigenvalues are

$$1 - \omega\lambda_1, \ldots, 1 - \omega\lambda_n.$$

Thus the spectral radius is given by

$$\rho(B) = \max_{i=1,\ldots,n} |1 - \omega\lambda_i|$$

The question is for which i we obtain

$$\min(\rho(B))?$$

For being s.p.d. (eigenvalues ordered, $\lambda_1 = \lambda_{min}$, $\lambda_n = \lambda_{max}$), we have

$$\rho(B) = \max(|1 - \omega\lambda_1|, |1 - \omega\lambda_n|) \tag{212}$$

Proposition 10.5 (Convergence Richardson)**.** *For A s.p.d. and ordered eigenvalues (real, positive), Richardon's method converges if and only if*

$$0 < \omega < \frac{2}{\lambda_n} \tag{213}$$

The optimal relaxation parameter is then given by

$$\omega_{opt} = \frac{2}{\lambda_1 + \lambda_n}$$

with the spectral radius

$$\rho(B) = \frac{\lambda_n - \lambda_1}{\lambda_1 + \lambda_n}$$

where $B = I - \omega_{opt} A$.

Proof. Let $0 < \omega < \frac{2}{\lambda_n}$ hold true. Then:

$$-1 < 1 - \omega\lambda_n \leq 1 - \omega\lambda_n < 1.$$

With (212), we obtain $\rho(B) < 1$ and therefore convergence. Let us now assume, we have convergence we want to show (213). In the case of convergence, for (212), it holds

$$1 > \rho(B) \geq |1 - \omega\lambda_n| \geq 1 - \omega\lambda_n.$$

Therefore: $\omega\lambda_n > 0$ since $\lambda_n > 0$. On the other hand:

$$-1 < -\rho(B) \leq -|1 - \omega\lambda_n| \leq 1 - \omega\lambda_n$$

showing $\omega\lambda_n < 2$, yielding (213). For the last part, consider the intersection $1 - \omega\lambda_1$ and $\omega\lambda_n - 1$ yielding the optimal value ω_{opt}. □

10.2.2 Jacobi and Gauss-Seidel

Further schemes require more work and we need to decompose the matrix A first:

$$A = L + D + U.$$

Here, L is a lower-triangular matrix, D a diagional matrix, and U an upper-triangular matrix. In more detail:

$$A = \underbrace{\begin{pmatrix} 0 & & \ldots & 0 \\ a_{21} & \ddots & & \\ \vdots & \ddots & \ddots & \\ a_{n1} & \ldots & a_{n,n-1} & 0 \end{pmatrix}}_{=:L} + \underbrace{\begin{pmatrix} a_{11} & & \ldots & 0 \\ & \ddots & & \\ & & \ddots & \\ 0 & \ldots & & a_{nn} \end{pmatrix}}_{=:D} + \underbrace{\begin{pmatrix} 0 & a_{12} & \ldots & a_{1n} \\ & \ddots & \ddots & \vdots \\ & & \ddots & a_{n-1,n} \\ 0 & \ldots & & 0 \end{pmatrix}}_{=:U}.$$

With this, we can now define two very important schemes:

Definition 10.6 (Jacobi method)**.** *To solve $Ax = b$ with $A = L + D + R$ let $x^0 \in \mathbb{R}^n$ be an initial guess. We iterate for $k = 1, 2, \ldots$*

$$x^k = x^{k-1} + D^{-1}(b - Ax^{k-1})$$

or in other words $J := -D^{-1}(L + R)$:

$$x^k = Jx^{k-1} + D^{-1}b.$$

Definition 10.7 (Gauß-Seidel method)**.** *To solve $Ax = b$ with $A = L + D + R$ let $x^0 \in \mathbb{R}^n$ be an initial guess. We iterate for $k = 1, 2, \ldots$*

$$x^k = x^{k-1} + (D + L)^{-1}(b - Ax^{k-1})$$

or in other words $H := -(D + L)^{-1}R$:

$$x^k = Hx^{k-1} + (D + L)^{-1}b.$$

To implement these two schemes, we provide the presentation in index-notation:

Theorem 10.8 (Index-notation of the Jacobi- and Gauß-Seidel methods)**.** *One step of the Jacobi method and Gauß-Seidel method, respectively, can be carried out in $n^2 + O(n)$ operations. For each step, in index-notation for each entry it holds:*

$$x_i^k = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1, j\neq i}^{n} a_{ij} x_j^{k-1} \right), \quad i = 1, \ldots, n,$$

i.e., (for the Gauss-Seidel method):

$$x_i^k = \frac{1}{a_{ii}} \left(b_i - \sum_{j<i} a_{ij} x_j^k - \sum_{j>i} a_{ij} x_j^{k-1} \right), \quad i = 1, \ldots, n.$$

10.3 Gradient descent

An alternative class of methods is based on so-called **descent** or **gradient** methods, which further improve the previously introduced methods. So far, we have:

$$x^{k+1} = x^k + d^k, \quad k = 1, 2, 3, \ldots$$

where d^k denotes the **direction** in which we go at each step. For instance:

$$d^k = D^{-1}(b - Ax^k), \quad d^k = (D + L)^{-1}(b - Ax^k)$$

for the Jacobi and Gauss-Seidel methods, respectively. To improve these kind of iterations, we have two possiblities:

- Introducing a relaxation (or so-called damping) parameter $\omega^k > 0$ (possibly adapted at each step) such that

$$x^{k+1} = x^k + \omega^k d^k,$$

and/or to improve the search direction d^k such that we reduce the error as best as possible. We restrict our attention to positive definite matrices as they appear in the discretization of elliptic PDEs studied previously in this section. A key point is another view on the problem by regarding it as a minimization problem for which $Ax = b$ is the first-order necessary condition and consequently the sought solution. Imagine for simplicity that we want to minimize $f(x) = \frac{1}{2}ax^2 - bx$. The first-order necessary condition is nothing else than the derivative $f'(x) = ax - b$. We find a possible minimum via $f'(x) = 0$, namely

$$ax - b = 0 \quad \Rightarrow \quad x = a^{-1}b, \quad \text{if } a \neq 0.$$

That is exactly the same how we would solve a linear matrix system $Ax = b$. By regarding it as a minimum problem we understand better the purpose of our derivations: How does minimizing a function $f(x)$ work in terms of an iteration? Well, we try to minimize f at each step k:

$$f(x^0) > f(x^1) > \ldots > f(x^k)$$

This means that the direction d^k (to determine $x^{k+1} = x^k + \omega^k d^k$) should be a descent direction. This idea can be applied to solving linear equation systems. We first define the quadratic form

$$Q(y) = \frac{1}{2}(Ay, y)_2 - (b, y)_2,$$

where $(\cdot, \cdot)$ is the Euclidian scalar product. Then, we can define

Algorithm 10.9 (Descent method - basic idea)**.** *Let $A \in \mathbb{R}^{n\times n}$ be positive definite and $x^0, b \in \mathbb{R}^n$. Then for $k = 0, 1, 2, \ldots$*

- *Compute d^k;*
- *Determine ω^k as minimum of $\omega^k = argmin\, Q(x^k + \omega^k d^k)$;*
- *Update $x^{k+1} = x^k + \omega^k d^k$.*

For instance d^k can be determined via the Jacobi or Gauss-Seidel methods.

Another possibility is the gradient method in which we use the gradient to obtain search directions d^k. This brings us to the gradient method:

Algorithm 10.10 (Gradient descent)**.** *Let $A \in \mathbb{R}^{n\times n}$ positive definite and the right hand side $b \in \mathbb{R}^n$. Let the initial guess be $x^0 \in \mathbb{R}$ and the initial search direction $d^0 = b - Ax^0$. Then $k = 0, 1, 2, \ldots$*

- *Compute the vector $r^k = Ad^k$;*
- *Compute the relaxation*
$$\omega^k = \frac{\|d_k\|_2^2}{(r^k, d^k)_2}$$
- *Update the solution vector $x^{k+1} = x^k + \omega^k d^k$.*
- *Update the search direction vector $d^{k+1} = d^k - \omega^k r^k$.*

One can show that the gradient method converges to the solution of the linear equation system $Ax = b$ (see for instance [114]).

Proposition 10.11 (Descent directions)**.** *Let $A \in \mathbb{R}^{n\times n}$ be symmetric positive definite. Then, two subsequent search directions d^k and d^{k+1} of the gradient descent scheme are orthogonal, i.e., $(d^k, d^{k+1}) = 0$.*

Proof. We have

$$d^{k+1} = d^k - \omega^k r^k = d^k - \frac{(d^k, d^k)}{(Ad^k, d^k)} Ad^k.$$

Therefore,

$$(d^{k+1}, d^k) = (d^k, d^k) - \frac{(d^k, d^k)}{(Ad^k, d^k)}(Ad^k, d^k) = (d^k, d^k) - (d^k, d^k) = 0.$$

□

10.4 Conjugate gradients (a Krylov space method)

This section is copy and paste from [114][Section 7.8] and translation from German into English. The relationship in Proposition 10.11 is only true for pairwise descent directions. In general, we have $d^k \not\perp d^{k+2}$. For this reason ,the gradient descent scheme converges slowly in most cases.

10.4.1 Formulation of the CG scheme

In order to enhance the performance of gradient descent, the conjugate gradient (CG) scheme was developed. Here, the search directions $\{d^0, \dots, d^{k-1}\}$ are pairwise orthogonal. The measure of orthogonality is achievend by using the A scalar product:

$$(Ad^r, d^s) = 0 \quad \forall r \neq s$$

At step k, we seek the approximation $x^k = x^0 + \sum_{i=0}^{k-1} \alpha_i d^i$ as the minimum of all $\alpha = (\alpha_0, \dots, \alpha_{k-1})$ with respect to $Q(x^k)$:

$$\min_{\alpha \in \mathbb{R}^k} Q\left(x^0 + \sum_{i=0}^{k-1} \alpha_i d^i\right) \quad = \quad \min_{\alpha \in \mathbb{R}^k} \left\{ \frac{1}{2} \left(Ax^0 + \sum_{i=0}^{k-1} \alpha_i A d^i, x^0 + \sum_{i=0}^{k-1} \alpha_i d^i \right) - \left(b, x^0 + \sum_{i=0}^{k-1} \alpha_i d^i \right) \right\}$$

The stationary point is given by

$$0 \stackrel{!}{=} \frac{\partial}{\partial \alpha_j} Q(x^k) = \left(Ax^0 + \sum_{i=0}^{k-1} \alpha_i A d^i, d^j \right) - (b, d^j) = -\left(b - Ax^k, d^j\right), \quad j = 0, \dots, k-1.$$

Therefore, the new residual $b - Ax^k$ is perpendicular to all search directions d^j for $j = 0, \dots, k-1$. The resulting linear equation system

$$(b - Ax^k, d^j) = 0 \quad \forall j = 0, \dots, k-1 \tag{214}$$

has the feature of Galerkin orthogonality, which we know as property of FEM schemes.

While constructing the CG method, new search directions should be linearly independent of the current d^j. Otherwise, the space would not become larger and consequently, the approximation cannot be improved.

Definition 10.12 (Krylov space)**.** *We choose an initial approximation $x^0 \in \mathbb{R}^n$ with $d^0 := b - Ax^0$ the Krylov space $K_k(d^0, A)$ such that*

$$K_k(d^0, A) := \operatorname{span}\{d^0, Ad^0, \dots, A^{k-1}d^0\}.$$

Here, A^k means the k-th power of A.

It holds:

Lemma 10.13. *Let $A^k d^0 \in K_k$. Then, the solution $x \in \mathbb{R}^n$ of $Ax = b$ is an element of the k-th Krylov space $K_k(d_0, A)$.*

Proof. Let K_k be given and $x^k \in x_0 + K_k$ the best approximation, which fulfills the Galerkin equation (214) Let $r^k := b - Ax^k$. Since

$$r^k = b - Ax^k = \underbrace{b - Ax^0}_{=d^0} + A \underbrace{(x^0 - x^k)}_{\in K_k} \in d^0 + AK_k$$

it holds $r^k \in K_{k+1}$. Supposing that $K_{k+1} \subset K_k$, we obtain $r^k \in K_k$. The Galerkin equation yields $r^k \perp K_k$, from which we obtain $r^k = 0$ and $Ax^k = b$. □

If the CG scheme aborts since it cannot find new search directions, the solution is found. Let us assume that the A-orthogonal search directions $\{d^0, d^1, \dots, d^{k-1}\}$ have been found, then we can compute the next CG approximation using the basis representation $x^k = x^0 + \sum \alpha_i d^i$ and employing the Galerkin equation:

$$\left(b - Ax^0 - \sum_{i=0}^{k-1} \alpha_i A d^i, d_j \right) = 0 \quad \Rightarrow \quad (b - Ax^0, d^j) = \alpha_j (Ad^j, d^j) \quad \Rightarrow \quad \alpha_j = \frac{(d^0, d^j)}{(Ad^j, d^j)}$$

The A-orthogonal basis $\{d^0, \ldots, d^{k-1}\}$ of the Krylov space $K_k(d^0, A)$ can be computed with the Gram-Schmidt procedure. However, this procedure has a high computational cost; see e.g., [114]. A better procedure is a two-step recursion formula, which is efficient and stable:

Lemma 10.14 (Two-step recursion formula)**.** *Let $A \in \mathbb{R}^{n\times n}$ symmetric positive definite and $x^0 \in \mathbb{R}^n$ and $d^0 := b - Ax^0$. Then, for $k = 1, 2, \ldots$, the iteration*

$$r^k := b - Ax^k, \quad \beta_{k-1} := -\frac{(r^k, Ad^{k-1})}{(d^{k-1}, Ad^{k-1})}, \quad d^k := r^k - \beta_{k-1} d^{k-1}$$

constructs an A-orthogonal basis with $(Ad^r, d^s) = 0$ for $r \neq s$. Here x^k in step k defines the new Galerkin solution $(b - Ax^k, d^j) = 0$ for $j = 0, \ldots, k-1$.

Proof. See [114]. □

We collect now all ingredients to construct the CG scheme. Let x^0 be an initial guess and $d^0 := b - Ax^0$ the resulting defect. Suppose that $K_k := \operatorname{span}\{d^0, \ldots, d^{k-1}\}$ and $x^k \in x^0 + K_k$ and $r^k = b - Ax^k$ have been computed. Then, we can compute the next iterate d^k according to Lemma 10.14:

$$\beta_{k-1} = -\frac{(r^k, Ad^{k-1})}{(d^{k-1}, Ad^{k-1})}, \quad d^k = r^k - \beta_{k-1} d^{k-1}. \tag{215}$$

For the new coefficient α_k in $x^{k+1} = x^0 + \sum_{i=0}^{k} \alpha_i d^i$ holds with testing in the Galerkin equation (214) with d^k:

$$\left(\underbrace{b - Ax^0}_{=d^0} - \sum_{i=0}^{k} \alpha_i A d^i, d^k \right) = (b - Ax^0, d^k) - \alpha_k (Ad^k, d^k) = (b - Ax^0 + \underbrace{A(x^0 - x^k)}_{\in K_k}, d^k) - \alpha_k (Ad^k, d^k).$$

That is

$$\alpha_k = \frac{(r^k, d^k)}{(Ad^k, d^k)}, \quad x^{k+1} = x^k + \alpha_k d^k. \tag{216}$$

This allows to compute the new defect r^{k+1}:

$$r^{k+1} = b - Ax^{k+1} = b - Ax^k - \alpha_k Ad^k = r^k - \alpha_k Ad^k \tag{217}$$

We summarize (215 − 217) and formulate the classical CG scheme:

Algorithm 10.15. *Let $A \in \mathbb{R}^{n\times n}$ symmetric positive definite and $x^0 \in \mathbb{R}^n$ and $r^0 = d^0 = b - Ax^0$ be given. Iterate for $k = 0, 1, \ldots$:*

1. $\alpha_k = \frac{(r^k, d^k)}{(Ad^k, d^k)}$
2. $x^{k+1} = x^k + \alpha_k d^k$
3. $r^{k+1} = r^k - \alpha_k Ad^k$
4. $\beta_k = \frac{(r^{k+1}, Ad^k)}{(d^k, Ad^k)}$
5. $d^{k+1} = r^{k+1} - \beta_k d^k$

Without round-off errors, the CG scheme yields after (at most) n steps the solution of a n-dimensional problem and is in this sense a direct method rather than an iterative scheme. However, in practice for huge n, the CG scheme is usually stopped earlier, yiedling an approximate solution.

Proposition 10.16 (CG as a direct method)**.** *Let $x^0 \in \mathbb{R}^n$ be any initial guess. Assuming no round-off errors, the CG scheme terminates after (at most) n steps with $x^n = x$. At each step, we have:*

$$Q(x^k) = \min_{\alpha \in \mathbb{R}} Q(x^{k-1} + \alpha d^{k-1}) = \min_{y \in x^0 + K_k} Q(y)$$

i.e.,

$$\|b - Ax^k\|_{A^{-1}} = \min_{y \in x^0 + K_k} \|b - Ay\|_{A^{-1}}$$

with the norm

$$\|x\|_{A^{-1}} = (A^{-1}x, x)_2^{\frac{1}{2}}.$$

Proof. That the CG scheme is a direct scheme follows from Lemma 10.13.

The iterate is given by

$$Q(x^k) = \min_{y \in x^0 + K_k} Q(y),$$

with is equivalent to (214). The ansatz

$$x^k = x^0 + \sum_{k=0}^{k-1} \alpha_k d^{k-1} = x^0 + \underbrace{y^{k-1}}_{\in K_{t-1}} + \alpha_{t-1} d^{k-1}$$

yields

$$(b - Ax^k, d^j) = (b - Ay^{k-1}, d_j) - \alpha_{t-1}(Ad^{k-1}, d^j) = 0 \quad \forall j = 0, \ldots, t-1$$

that is

$$(b - Ay^{k-1}, d_j) = 0 \; \forall j = 0, \ldots, t-2,$$

and therefore $y^{k-1} = x^{k-1}$ and

$$Q(x^k) = \min_{\alpha \in \mathbb{R}} Q(x^{k-1} + \alpha d^{k-1}).$$

Finally, employing symmetry $A = A^T$, we obtain:

$$\begin{aligned} \|b - Ay\|_{A^{-1}}^2 = (A^{-1}[b - Ay], b - Ay)_2 &= (Ay, y)_2 - (A^{-1}b, Ay)_2 - (y, b)_2 \\ &= (Ay, y)_2 - 2(b, y)_2, \end{aligned}$$

i.e., the relationship $\|b - Ay\|_{A^{-1}}^2 = 2Q(y)$. □

Remark 10.17 (The CG scheme as direct method)**.** *In Section 10.8 we solve the linear equation system resulting from Poisson's problem with several schemes. Using the CG scheme without preconditioning, we see that is terminates with an iteration number $M < N = DoFs$. For instance for $N = 25 DoFs$, we have $M = 3$ iterations. For larger N it is of course interesting to terminate the CG scheme earlier than machine precision, which - as previously stated - is the main interest in general.*

Remark 10.18 (The CG scheme as iterative scheme)**.** *As previosously mentioned, in pratice the CG scheme is (always) used as iterative method rather than a direct method. Due to round-off errors the search directions are never* 100% *orthogonal.*

10.4.2 Convergence analysis of the CG scheme

We now turn our attention to the convergence analysis, which is a nontrivial task. The key is the following characterization of one iteration $x^k = x^0 + K_k$ by

$$x^k = x^0 + p_{k-1}(A)d^0,$$

where $p_{k-1} \in P_{k-1}$ is a polynomial in A:

$$p_{k-1}(A) = \sum_{i=0}^{k-1} \alpha_i A^i$$

The characterization as minimization of Proposition 10.16 can be written as:

$$\|b - Ax^k\|_{A^{-1}} = \min_{y \in x^0 + K_k} \|b - Ay\|_{A^{-1}} = \min_{q \in P_{k-1}} \|b - Ax^0 - Aq(A)d^0\|_{A^{-1}}.$$

When we employ the $\|\cdot\|_A$ norm, we obtain with $d^0 = b - Ax^0 = A(x - x^0)$

$$\|b - Ax^k\|_{A^{-1}} = \|x - x^k\|_A = \min_{q \in P_{k-1}} \|(x - x^0) - q(A)A(x - x^0)\|_A,$$

that is

$$\|x - x^k\|_A = \min_{q \in P_{k-1}} \|[I - q(A)A](x - x^0)\|_A.$$

In the sense of the best approximation property (recall e.g., Section 8.7), we can formulate this task as:

$$p \in P_{k-1}: \quad \|[I - p(A)A](x - x^0)\|_A = \min_{q \in P_{k-1}} \|[I + q(A)A](x - x^0)\|_A. \tag{218}$$

The characterization as best approximation is key in the convergence analysis of the CG scheme. Let $q(A)A \in P_k(A)$. We seek a polynomial $q \in P_k$ with $q(0) = 1$, such that

$$\|x^k - x\|_A \leq \min_{q \in P_k,\, q(0)=1} \|q(A)\|_A \|x - x^0\|_A. \tag{219}$$

The convergence of the CG method is related to the fact whether we can construct a polynomial $q \in P_k$ with $p(0) = 1$ such that the A norm is as small as possible. First, we have:

Lemma 10.19 (Bounds for matrix polynomials)**.** *Let $A \in \mathbb{R}^{n \times n}$ symmetric positive definite with the eigenvalues $0 < \lambda_1 \leq \cdots \leq \lambda_n$, and $p \in P_k$ a polynomials with $p(0) = 1$:*

$$\|p(A)\|_A \leq M, \quad M := \min_{p \in P_k,\, p(0)=1} \sup_{\lambda \in [\lambda_1, \lambda_n]} |p(\lambda)|.$$

Proof. See [114]. □

Employing the previous result and the error estimate (219), we can now derive a convergence result for the CG scheme.

Proposition 10.20 (Convergence of the CG scheme)**.** *Let $A \in \mathbb{R}^{n \times n}$ be symmetric positive definite. Let $b \in \mathbb{R}^n$ a right hand side vector and let $x^0 \in \mathbb{R}^n$ be an initial guess. Then:*

$$\|x^k - x\|_A \leq 2 \left(\frac{1 - 1/\sqrt{\kappa}}{1 + 1/\sqrt{\kappa}} \right)^k \|x^0 - x\|_A, \quad k \geq 0,$$

with the spectral condition $\kappa = \mathrm{cond}_2(A)$ of the matrix A.

Remark 10.21. *We see immediately that a large condition number $\kappa \gg 1$ yields*

$$\frac{1 - 1/\sqrt{\kappa}}{1 + 1/\sqrt{\kappa}} \to 1$$

and deteriorates significantly the convergence rate of the CG scheme. This is the key reason why preconditioners of the form $P^{-1} \approx A^{-1}$ are introduced that re-scale the system; see Section 10.5:

$$\underbrace{P^{-1}A}_{\approx I} x = P^{-1}b.$$

Computations to substantiate these findings are provided in Section 10.8.

Proof. Of Prop. 10.20.
From the previous lemma and the estimate (219) it follows

$$\|x^k - x\|_A \leq M \|x^0 - x\|_A$$

with

$$M = \min_{q \in P_k,\, q(0)=1} \max_{\lambda \in [\lambda_1, \lambda_n]} |q(\lambda)|.$$

We have to find a sharp estimate of the size of M. That is to say, we seek a polynomial $q \in P_k$ which takes at the origin the value 1, i.e., $q(0) = 1$ and which simultaneously has values near 0 in the maximum norm in the interval $[\lambda_1, \lambda_n]$. To this end, we work with the Tschebyscheff approximation (see e.g., [114] and references therein for the original references). We seek a best approximation $p \in P_k$ of the zero function on $[\lambda_1, \lambda_n]$. Such a polynomial should have the property $p(0) = 1$. For this reason, the trivial solution $p = 0$ is not valid. A Tschebyscheff polynomial reads:

$$T_k = \cos\big(k \arccos(x)\big)$$

and has the property:

$$2^{-k-1} \max_{[-1,1]} |T_k(x)| = \min_{\alpha_0, \ldots, \alpha_{k-1}} \max_{[-1,1]} |x^k + \sum_{i=0}^{k-1} \alpha_i x^k|.$$

We choose now the transformation:

$$x \mapsto \frac{\lambda_n + \lambda_1 - 2t}{\lambda_n - \lambda_1}$$

and obtain with

$$p(t) = T_k\left(\frac{\lambda_n + \lambda_1 - 2t}{\lambda_n - \lambda_1}\right) T_k\left(\frac{\lambda_n + \lambda_1}{\lambda_n - \lambda_1}\right)^{-1}$$

a polynomial of degree k, which is minimal on $[\lambda_1, \lambda_n]$ and can be normalized by

$$p(0) = 1.$$

It holds:

$$\sup_{t \in [\lambda_1, \lambda_n]} |p(t)| = T_k\left(\frac{\lambda_n + \lambda_1}{\lambda_n - \lambda_1}\right)^{-1} = T_k\left(\frac{\kappa + 1}{\kappa - 1}\right)^{-1} \tag{220}$$

with the spectral condition:

$$\kappa := \frac{\lambda_n}{\lambda_1}.$$

We now employ the Tschebyscheff polynomials outside of $[-1, 1]$:

$$T_n(x) = \frac{1}{2}\Big[(x + \sqrt{x^2 - 1})^n + (x - \sqrt{x^2 - 1})^n\Big]$$

For $x = \frac{\kappa+1}{\kappa-1}$, it holds:

$$\frac{\kappa + 1}{\kappa - 1} + \sqrt{\left(\frac{\kappa + 1}{\kappa - 1}\right)^2 - 1} = \frac{\kappa + 2\sqrt{\kappa} + 1}{\kappa - 1} = \frac{\sqrt{\kappa} + 1}{\sqrt{\kappa} - 1}$$

and therefore

$$\frac{\kappa + 1}{\kappa - 1} - \sqrt{\left(\frac{\kappa + 1}{\kappa - 1}\right)^2 - 1} = \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}.$$

Using this relationship, we can estimate (220):

$$T_k\left(\frac{\kappa + 1}{\kappa - 1}\right) = \frac{1}{2}\Big[\left(\frac{\sqrt{\kappa} + 1}{\sqrt{\kappa} - 1}\right)^k + \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^k\Big] \geq \frac{1}{2}\left(\frac{\sqrt{\kappa} + 1}{\sqrt{\kappa} - 1}\right)^k$$

It follows that

$$\sup_{t \in [\lambda_1, \lambda_n]} T_k\left(\frac{\kappa + 1}{\kappa - 1}\right)^{-1} \leq 2\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^k = 2\left(\frac{1 - \frac{1}{\sqrt{\kappa}}}{1 + \frac{1}{\sqrt{\kappa}}}\right)^k.$$

This finishes the proof. □

10.5 Preconditioning

This section is mostly copy and paste from [114][Section 7.8.1] and translation from German into English.

The rate of convergence of iteratice schemes depends on the condition number of the system matrix. For instance, we recall that for second-order operators (such as Laplace) we have a dependence on the mesh size $O(h^{-2}) = O(N)$ (in 2D) (see Section 16 for the relation between h and N). For the CG scheme it holds:

$$\rho_{CG} = \frac{1 - \frac{1}{\sqrt{\kappa}}}{1 + \frac{1}{\sqrt{\kappa}}} = 1 - \frac{2}{\sqrt{\kappa}} + O\left(\frac{1}{\kappa}\right).$$

Preconditioning reformulates the original system with the goal of obtaining a moderate condition number for the modified system. To this end, we seek a matrix $P \in \mathbb{R}^{n\times n}$ such that

$$P \approx A^{-1}.$$

On the other hand

$$P \approx I,$$

such that the construction of P is not too costly. Obviously, these are two conflicting requirements.

When A is symmetric (for instance Poisson problem), P can be composed of two matrices:

$$P = KK^T.$$

Then:

$$Ax = b \quad \Leftrightarrow \quad \underbrace{K^{-1}A(K^T)^{-1}}_{=:\tilde{A}} \underbrace{K^T x}_{=:\tilde{x}} = \underbrace{K^{-1}b}_{=:\tilde{b}},$$

which is

$$\tilde{A}\tilde{x} = \tilde{b}.$$

In the case of

$$\mathrm{cond}_2(\tilde{A}) \ll \mathrm{cond}_2(A)$$

and if the application of K^{-1} is cheap, then the consideration of a preconditioned system $\tilde{A}\tilde{x} = \tilde{b}$ yields a much faster solution of the iterative scheme. The condition $P = KK^T$ is necessary such that the matrix $\tilde{A}$ keeps its symmetry.

The preconditioned CG scheme (PCG) can be formulated as:

Algorithm 10.22. *Let $A \in \mathbb{R}^{n\times n}$ symmetric positive definite and $P = KK^T$ a symmetric preconditioner. Choosing an initial guess $x^0 \in \mathbb{R}^n$ yields:*

1. $r^0 = b - Ax^0$
2. $Pp^0 = r^0$
3. $d^0 = p^0$
4. *For* $k = 0, 1, \ldots$
   a) $\alpha_k = \frac{(r^k, d^k)}{(Ad^k, d^k)}$
   b) $x^{k+1} = x^k + \alpha_k d^k$
   c) $r^{k+1} = r^k - \alpha_k A d^k$
   d) $Pp^{k+1} = r^{k+1}$
   e) $\beta_k = \frac{(r^{k+1}, p^{k+1})}{(r^k, g^k)}$
   f) $d^{k+1} = p^{k+1} + \beta_k d^k$

At each step, we have as additional cost the application of the preconditioner P. We recall that P allows the decomposition into K and K^T even if they are not explicitely used.

Typical preconditioners are:

- *Jacobi preconditioning*
  We choose $P \approx D^{-1}$, where D is the diagonal part of A. It holds

  $$D = D^{\frac{1}{2}}(D^{\frac{1}{2}})^T,$$

  which means that for $D_{ii} > 0$, this preconditioner is admissible. For the preconditioned matrix, it holds

  $$\tilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \quad \Rightarrow \quad \tilde{a}_{ii} = 1$$

- *SSOR preconditioning*

  The SSOR scheme is a symmetric variant of the SOR method (successive over-relaxation) and is based on the decomposition:

  $$P = (D + \omega L)D^{-1}(D + \omega R) = \underbrace{(D^{\frac{1}{2}} + \omega L D^{-\frac{1}{2}})}_{K}\underbrace{(D^{\frac{1}{2}} + \omega D^{-\frac{1}{2}} R)}_{=K^T}.$$

  For instance, for the Poisson problem, we can find an optimal ω (which is a non-trivial task) such that

  $$\mathrm{cond}_2(\tilde{A}) = \sqrt{\mathrm{cond}_2(A)}$$

  can be shown. Here, the convergence improves significantly. The number of necessary steps to achieve a given error reduction by a factor of ϵ improves to

  $$t_{CG}(\epsilon) = \frac{\log(\epsilon)}{\log(1 - \kappa^{-\frac{1}{2}})} \approx -\frac{\log(\epsilon)}{\sqrt{\kappa}}, \quad \tilde{t}_{CG}(\epsilon) = \frac{\log(\epsilon)}{\log(1 - \kappa^{-\frac{1}{4}})} \approx \frac{\log(\epsilon)}{\sqrt[4]{\kappa}}.$$

  Rather than having 100 steps, we only need 10 steps for instance, in case an optimal ω can be found.

10.6 GMRES - generalized minimal residual method

The iterative GMRES algorithm [115, 116] is suited for solving nonsymmetric linear systems. Let $A \in \mathbb{R}^{n \times n}$ be a regular matrix, but not necessarily symmetric. We demonstrate first an option, which turns out to be not feasible when n is large. A symmetric version of the problem

$$Ax = b$$

can be achieved by multiplication with A^T:

$$A^T A x = A^T b.$$

The matrix $B = A^T A$ is positive definite since

$$(Bx, x)_2 = (A^T A x, x)_2 = (Ax, Ax)_2 = \|Ax\|_2.$$

In principle, we could now apply the CG scheme to $A^T A$. Instead of one matrix-vector multiplication, we would need two such multiplications per step. However, using $A^T A$, the convergence rate will deteriorate since

$$\kappa(B) = \mathrm{cond}_2(A^T A) = \mathrm{cond}_2(A)^2.$$

For this reason, the CG scheme is not really an option.

10.6.1 Pure GMRES

Let us now prepare the ingredients for the GMRES scheme. The main part is Arnoldi's method [7] in which the (modified) Gram-Schmidt method (see Numerik 1, e.g. [114] for the modified Gram-Schmidt see e.g., [115]) is used for orthonormalization of the basis $\{v_1, \ldots, v_k\}$ of the Krylov space $K_k = \mathrm{span}\{v_1, Av_1, \ldots, A^{k-1}v_1\}$.

Algorithm 10.23 (Arnoldi). *Choose some initial vector v_1 with $\|v_1\| = 1$. Iterate for $j = 1, 2, \ldots,$ do*

1. $h_{i,j} = (Av_j, v)_2, \quad i = 1, 2, \ldots, j$
2. $\hat{v}_{j+1} = Av_j - \sum_{i=1}^{j} h_{i,j} v_i$
3. $h_{j+1,j} = \|\hat{v}_{j+1}\|$
4. $v_{j+1} = \hat{v}_{j+1}/h_{j+1,j}$

We then define

$$V_k = \{v_1, \ldots, v_k\} \in \mathbb{R}^{n \times k}$$

and $H_k = V_k^T A V_k \in \mathbb{R}^{k \times k}$ is the upper Hessenberg matrix, with entries h_{ij} just computed by Arnoldi's algorithm.

Recall the task: Solve

$$Ax = b, \quad A \in \mathbb{R}^{n \times n}$$

Using a Galerkin method with an l_2-orthonormal basis (produced by Arnoldi's algorithm), we seek x_k of the form

$$x_k = x_0 + z_k$$

where x_0 is some initial guess and z_k comes from the Krylov space $K_k = \text{span}\{r_0, Ar_0, \ldots, A^{k-1} r_0\}$ with the usual initial residual $r_0 = b - Ax_0$.

In order to prepare for GMRES, we assume that after k steps of Arnoldi's algorithm, we have

- an orthonormal system $V_{k+1} = \{v_1, \ldots, v_{k+1}\}$
- a matrix $\bar{H}_k \in \mathbb{R}^{(k+1) \times k}$ with the non-zero elements h_{ij}.

We have

$$AV_k = V_{k+1} \bar{H}_k. \tag{221}$$

With this, we are interested in solving the following least squares problem:

$$\min_{z \in K_k} \|b - A[x_0 + z]\| = \min_{z \in K_k} \|r_0 - Az\|. \tag{222}$$

Introducing the variable $z = V_k y$ we obtain the following minimization problem:

$$\min J(y) = \min \|\beta v_1 - AV_k y\|, \quad \beta = \|r_0\|.$$

We notice that (using (221))

$$b - Ax = b - A(x_0 + V_k y) = r_0 - AV_k y = \beta v_1 - V_{k+1} \bar{H}_k y = V_{k+1}(\beta e_1 - \bar{H}_k y)$$

This brings us to

$$J(y) = \|V_{k+1}[\beta e_1 - \bar{H}_k y]\|$$

with the unit vector $e_1 \in \mathbb{R}^{(k+1) \times (k+1)}$. Since V_{k+1} is orthonormal we have

$$J(y) = \|\beta e_1 - \bar{H}_k y\|, \quad y \in \mathbb{R}^k.$$

The solution to (222) is given by

$$x_k = x_0 + V_k y_k$$

where y_k minimizes $J(y)$. This minimizer y_k is inexpensive since it requires to solve a $(k+1) \times k$ least-squares problem, where k is typically small.

Algorithm 10.24. *GMRES:*

1. *Choose the initial guess x_0. Compute $r_0 = b - Ax_0$ and $v_1 = r_0/\|r_0\|$.*
2. *Iterate: For $j = 1, 2, \ldots, k, \ldots$ until some stopping criterion:*

a) $h_{i,j} = (Av_j, v_i)_2, \quad i = 1, 2, \ldots, j$

b) $\hat{v}_{j+1} = Av_j - \sum_{i=1}^{j} h_{i,j} v_i$

c) $h_{j+1,j} = \|\hat{v}_{j+1}\|$. *If* $h_{j+1,j} = 0$ *set* $k := j$ *and go to Step 3 (Hessenberg matrix)*

d) $v_{j+1} = \hat{v}_{j+1}/h_{j+1,j}$

3. *Define the* $(k+1) \times k$ *Hessenberg matrix* $\bar{H}_k = \{h_{ij}\}_{1 \le ik+1, 1 \le j \le k}$;

4. *Compute minimizer* y_k *of* $J(y)$;

5. *Set as approximate solution* $x_k = x_0 + V_k y_k$.

Unfortunately, for increasing k the storage of vectors becomes an issue and increases like k. Moreover, the arithmetic cost increases as $1/2k^2 n$. Consequently, there exists a restared version, say after m steps, denoted by GMRES(m):

Algorithm 10.25. *GMRES(m):*

1. *Choose the initial guess* x_0. *Compute* $r_0 = b - Ax_0$ *and* $v_1 = r_0/\|r_0\|$.

2. *Iterate: For* $j = 1, 2, \ldots, m$ *do:*

a) $h_{i,j} = (Av_j, v_i)_2, \quad i = 1, 2, \ldots, j$

b) $\hat{v}_{j+1} = Av_j - \sum_{i=1}^{j} h_{i,j} v_i$

c) $h_{j+1,j} = \|\hat{v}_{j+1}\|$

d) $v_{j+1} = \hat{v}_{j+1}/h_{j+1,j}$

3. *Set as approximate solution* $x_m = x_0 + V_m y_k$ *where* y_m *minimizes* $\|\beta e_1 - \bar{H}_m y\|$ *for* $y \in \mathbb{R}^m$.

4. *Restart:*

- *Compute* $r_m = b - Ax_m$; *if satisfied, stop*
- *else compute* $x_0 := x_m, r_1 := r_m/\|r_m\|$ *and go to step 2.*

Remark 10.26. *Of course in both versions, still the minimization problem* $J(y)$ *must be solved, which can be done by the QR method. For more details, we refer the reader to [115].*

10.6.2 Left-preconditioned GMRES

The left-preconditioned scheme reads

$$P^{-1}Ax = P^{-1}b.$$

Then, we have

Algorithm 10.27. *Left-preconditioned GMRES:*

1. *Compute* $r_0 = P^{-1}(b - Ax_0)$, $\beta = \|r_0\|_2$, *and* $v_1 = r_0/\beta$

2. *For* $j = 1, \ldots, k$ *do*

a) Compute $w := P^{-1}Av_j$

b) For $i = 1, \ldots, j$ *do*

i. $h_{i,j} := (w, v_i)$

ii. $w := w - h_{i,j} v_i$

c) Compute $h_{j+1,j} = \|w\|_2$ *and* $v_{j+1} = w/h_{j+1,j}$

3. *Define* $V_k := \{v_1, \ldots, v_k\}$ *and* $\bar{H}_k = \{h_{i,j}\}_{1 \le i \le j+1, 1 \le j \le k}$

4. *Compute* $y_k = argmin_y \|\beta e_1 - \bar{H}_k y\|_2$

5. *Set* $x_k = x_0 + V_k y_k$

6. *If satisfied stop (solution found), else set* $x_0 := x_k$ *and go to Step 1.*

10.6.3 Right-preconditioned GMRES

The right-preconditioned scheme reads

$$AP^{-1}u = b, \quad u = Px.$$

Then, we have

Algorithm 10.28. *Right-preconditioned GMRES:*

1. *Compute* $r_0 = b - Ax_0$, $\beta = \|r_0\|_2$, *and* $v_1 = r_0/\beta$
2. *For* $j = 1, \ldots, k$ *do*
   a) *Compute* $w := AP^{-1}v_j$
   b) *For* $i = 1, \ldots, j$ *do*
      i. $h_{i,j} := (w, v_i)$
      ii. $w := w - h_{i,j}v_i$
   c) *Compute* $h_{j+1,j} = \|w\|_2$ *and* $v_{j+1} = w/h_{j+1,j}$
3. *Define* $V_k := \{v_1, \ldots, v_k\}$ *and* $\bar{H}_k = \{h_{i,j}\}_{1 \le i \le j+1, 1 \le j \le k}$
4. *Compute* $y_k = argmin_y \|\beta e_1 - \bar{H}_k y\|_2$
5. *Set* $x_k = x_0 + P^{-1}V_k y_k$
6. *If satisfied stop (solution found), else set* $x_0 := x_k$ *and go to Step 1.*

The difference to the left-preconditioned version is that the residual norm is taken with respect to the initial system $b - Ax_k$ *(without* P^{-1}*). The reason is that the algorithm obtains the residual implicitly* $b - Ax_k = b - AP^{-1}u_k$*. Moreover, the right-preconditioned version can be extended to FGMRES (where 'F' stands for 'flexible') in which the preconditioner* P^{-1} *can be changed at each iteration step [115].*

10.7 Geometric multigrid methods

In this section, we follow closely [13][Chapter 10]. Excellent descriptions including historical notes can be found in Braess [24] and the famous book from Hackbusch [67]. Also [115] is recommended.

10.7.1 Motivation

For elliptic problems, the linear systems are large, sparse and symmetric positiv definite (s.p.d.). These properties already were necessary to use the CG method for the iterative solution. However, the convergence properties depend on the condition number. We recall again that the usual stiffness matrix behaves as (see Section 10.1)

$$\kappa(A) = \frac{\lambda_{max}}{\lambda_{min}} = O(h^{-2}).$$

Example 10.29. *Referring forward to Section 10.7.3, we obtain for instance for* $h = 0.5, 0.25, 0.125$*:*

$$\kappa_{h=0.5}(A) = 8$$
$$\kappa_{h=0.25}(A) = 54$$
$$\kappa_{h=0.125}(A) = 246$$

reflecting that the condition number grows with $O(h^{-2})$*.*

The work amount in terms of arithmetic operations for often-used methods is listed in Table 1. Therein, we already see that the CG method may perform extremely well; in particular in 3d.

Table 1: Operations for solving $Az = b$ with $A \in \mathbb{R}^{n\times n}$ being large, sparse, and s.p.d.

Scheme	$d = 2$	$d = 3$
Gauss (direct)	n^3	n^3
Banded-Gauss (direct)	n^2	$n^{7/3}$
Jacobi (iterative)	n^2	$n^{5/3}$
Gauss-Seidel (iterative)	n^2	$n^{5/3}$
CG	$n^{3/2}$	$n^{4/3}$
SOR with opt. ω	$n^{3/2}$	$n^{4/3}$
Multigrid	n	n

Multigrid methods can solve linear systems with optimal complexity; namely n. Less than n is not possible since each entry has to be looked up at least once.

Algorithm 10.30 (Basic steps of multigrid)**.** *The basic procedure of geometric multigrid is:*

- *Solve a given PDE on a hierarchy of meshes (grids);*
- *Use on each mesh (grid) a (simple) iterative solver, e.g., Jacobi or Gauss-Seidel, the so-called* ***smoother****. The smoother will damp quickly high oscillatory parts of errors on each respective mesh (grid) level;*
- *Transfer (****restriction****) solutions from fine grids to coarser grids and collect finally (****prolongation****) all parts to a final solution on the finest grid.*

10.7.2 Smoothing property of Richardson iteration

In this first section, we explain the motivation why to solve on a hierarchy of meshes using a simple iterative solver. Let

$$Ax = b$$

be given. Richardson iteration (Numerik 1 [114]) then reads:

$$x^{k+1} = x^k + \omega(b - Ax^k)$$

with a relaxation parameter ω. We know that Richardson converges when $0 < \omega < 2/\lambda_{max}(A)$ (see Section 10.2.1). Since A is s.p.d. the spectrum of eigenvalues can be ordered as

$$\sigma(A) = \{\lambda_{min}(A) = \lambda_1, \lambda_2, \ldots, \lambda_n = \lambda_{max}(A)\}$$

with

$$0 < \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n.$$

We can now write the iteration error $e^{k+1} = x - x^{k+1}$. Specifically:

$$\begin{aligned} e^{k+1} &= x - x^{k+1} \\ &= x - x^k - \omega(b - Ax^k) \quad \text{(Richardson iteration function)} \\ &= (I - \omega A)(x - x^k) \\ &= (I - \omega A)e^k \\ &= Me^k. \end{aligned}$$

The matrix $M = (I - \omega A)$ is the so-called iteration matrix. In Numerik 1, we have analyzed under which conditions for M we obtain convergence. Since we A is s.p.d. we can perform an eigenvalue decomposition. Let (λ_i, z_i) an eigenpair of A, where λ_i is the eigenvalue and z_i the eigenvector. We recall that $Az_i = \lambda_i z_i$ is the eigenvalue equation. We apply this idea to the Me^k:

$$Mz_i = (I - \omega A)z_i = (1 - \omega\lambda_i)z_i.$$

We recall from linear algebra that the $z_i, i = 1, \ldots, n$ form a basis of $\mathbb{R}^n$. Consequently, any vector e^k can be written as linear combination with the help of the basis vectors. Thus:

$$Me^k = M\sum_{i=1}^{n} a_i z_i = \sum_{i=1}^{n} a_i(1 - \omega\lambda_i)z_i,$$

where $a_i \in \mathbb{R}$ are the coefficients. Now we have a relation between the error e and the reduction factor $\eta := (1-\omega\lambda_i)$. It is clear that the error depends on the size of $(1-\omega\lambda_i)$. For the special choice $\omega = 1/\lambda_n < 2/\lambda_n$, we observe the reduction factor:

$$\begin{aligned} 1 - \omega\lambda_i = 1 - \frac{\lambda_i}{\lambda_n} \quad &\to 0 \quad \text{when } i \text{ is large, i.e., } \lambda_i \approx \lambda_n \\ 1 - \omega\lambda_i = 1 - \frac{\lambda_i}{\lambda_n} \quad &\to 1 \quad \text{when } i \text{ is small, i.e., } \lambda_i \ll \lambda_n \end{aligned}$$

We remember from Numerik 1 that the first situation is much better, because we have a fast error reduction. What does this observation mean? First, large eigenvalues with $\lambda_i \approx \lambda_n$ are damped (smoothed) quickly in the iteration error. Second, smaller eigenvalues are not damped quickly. However, smaller eigenvalues become larger on coarser grids (smaller n). And here we are: we use a hierachy of meshes in order to damp all eigenvalues efficiently.

10.7.2.1 Practical procedure when λ_n not known In practice λ_n is often not known. In this case, an upper bound can be obtained with Gershgorin circles (see Numerik 2; e.g., [117]), such that we have $\lambda_n \leq \gamma$. We then set $\omega = \frac{1}{\gamma}$. Clearly, we then have still convergence since

$$\frac{1}{\gamma} \leq \frac{1}{\lambda_n} < \frac{2}{\lambda_n}.$$

Then,

$$\eta = (1 - \omega\lambda_i) = \left(1 - \frac{\lambda_i}{\gamma}\right). \tag{223}$$

Slowest convergence is obtained for $i = 1$ with $\lambda_1/\gamma \ll 1$ and for $i = n$ we have fast convergence with $\lambda_n/\gamma \approx 1$.

10.7.3 Eigenvalues and eigenvectors for 1D Poisson and mesh independence

For Poisson in 1D,

$$\begin{aligned} -u''(x) &= f \quad \text{in } \Omega = (0,1) \\ u(0) = u(1) &= 0 \end{aligned}$$

This problem is discretized on the mesh $\mathcal{T}_h = \{x_i\}_{i=0,\dots,n+1}$. Thus, we have $n+2$ Dofs including the boundary dofs x_0 and x_{n+1} and n inner degrees of freedom $x_1, \dots, x_n$. The mesh size is $h = 1/(n+1)$. We have our well known matrix (by having h^2 on the right hand side and excluding the boundary points):

$$A = \begin{pmatrix} -1 & 2 & -1 & 0 & 0 \\ & \ddots & & & \\ 0 & -1 & 2 & -1 & 0 \\ & \ddots & & & \\ 0 & 0 & -1 & 2 & -1 \end{pmatrix} \in \mathbb{R}^{n\times n}, \qquad b = h^2 \begin{pmatrix} f(x_1) \\ \vdots \\ \vdots \\ \vdots \\ f(x_n) \end{pmatrix} \in \mathbb{R}^n.$$

We recall the eigenvalue equation

$$Aw = \lambda w.$$

The eigenvalues are

$$\lambda_i = 4\sin^2\frac{\theta_i}{2}, \quad \theta_i = \frac{i\pi}{n+1} = ih\pi, \quad i = 1, \dots n$$

and the corresponding eigenvectors:

$$w_i = \begin{pmatrix} \sin(\theta_i) \\ \sin(2\theta_i) \\ \vdots \\ \sin(n\theta_i) \end{pmatrix} \quad \in \mathbb{R}^n, \quad i = 1, \dots, n.$$

We now use (223) and estimate with Gershgorin circles that $\gamma = 4$ (clear since $|x-2| \leq |-1| + |-1| = 2$). Then,

$$\eta = \left(1 - \frac{\lambda_i}{4}\right) = \left(1 - \frac{4\sin^2\frac{\theta_i}{2}}{4}\right) = \left(1 - \sin^2\frac{i\pi h}{2}\right) \tag{224}$$

- For $i = 1$ (smallest eigenvalue), we obtain

  $$\left(1 - \sin^2\frac{\pi h}{2}\right) \approx 1 - \left(\frac{\pi h}{2}\right) = 1 - O(h^2).$$

  It is clear that for $h \to 0$, we obtain

  $$1 - O(h^2) \approx 1.$$

  In particular, this result is h-dependent: the smaller h, the worse.

- For $i = n$ (largest eigenvalue), we have

$$\left(1 - \sin^2 \frac{n\pi h}{2}\right) = \cos^2 \frac{n\pi}{2(n+1)} \leq \frac{1}{2},$$

  which holds for any $n = 1, 2, \ldots$ and therefore the bound $1/2$ is **independent of** n and therefore h.

- $i > n/2$:

$$\left(1 - \sin^2 \frac{i\pi h}{2}\right) = \cos^2 \frac{i\pi}{2(n+1)} \leq \frac{1}{2},$$

  independely of n (thus h).

Low frequency eigenvectors are obtained for small i. High frequency eigenvectors are for $i \approx n$. Whether an eigenvalue is low frequent or not depends on the mesh size h (and consequently on n). This means:

- High frequency errors with $i > n/2$ are damped quickly with Richardson;
- Low frequency errors with $i \leq i/2$ are damped slowly with Richardson.

Consequently, if we transfer low frequency errors to coarser meshes, they become high frequent and we could again damp them with Richardson. The realization requires consequently are hierarchy of meshes.

10.7.4 Numerical example of eigenvalues and eigenvectors for 1D Poisson

We demonstrate our previous descriptions with a numerical example on three different (hierarchical) meshes. The starting point is Section 6.7 with the matrix $A = \frac{1}{h^2}(\ldots)$. We use octave and specifically the function

```
[V,lambda] = eig(A)
```

We use on $\Omega = (0, 1)$ three mesh sizes $h = 0.5, 0.25, 0.125$ corresponding to $n = 1, 3, 7$ (excluding the boundary values; thus each time $x_0, \ldots, x_{n+1}$, thus $n + 2$ nodel points), respectively[8]. Starting on the finest mesh, we obtain:

```
A2 =

     1     0     0     0     0     0     0     0     0
   -64   128   -64     0     0     0     0     0     0
     0   -64   128   -64     0     0     0     0     0
     0     0   -64   128   -64     0     0     0     0
     0     0     0   -64   128   -64     0     0     0
     0     0     0     0   -64   128   -64     0     0
     0     0     0     0     0   -64   128   -64     0
     0     0     0     0     0     0   -64   128   -64
     0     0     0     0     0     0     0     0     1

V =

   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000   0.52743   0.00000
   0.19134  -0.35355   0.46194   0.50000  -0.19134  -0.35355   0.46194   0.48112   0.07816
  -0.35355   0.50000  -0.35355  -0.00000  -0.35355  -0.50000   0.35355   0.42729   0.15511
   0.46194  -0.35355  -0.19134  -0.50000  -0.46194  -0.35355  -0.19134   0.36679   0.22962
  -0.50000  -0.00000   0.50000  -0.00000  -0.50000   0.00000  -0.50000   0.30055   0.30055
   0.46194   0.35355  -0.19134   0.50000  -0.46194   0.35355  -0.19134   0.22962   0.36679
  -0.35355  -0.50000  -0.35355   0.00000  -0.35355   0.50000   0.35355   0.15511   0.42729
   0.19134   0.35355   0.46194  -0.50000  -0.19134   0.35355   0.46194   0.07816   0.48112
   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000   0.52743
```

[8]For the correspondance of nodal points n and the mesh size h, we refer the reader to Section 16.1.1.

```
lambda =

   246.2566         0         0         0         0         0         0         0         0
         0   218.5097         0         0         0         0         0         0         0
         0         0   176.9835         0         0         0         0         0         0
         0         0         0   128.0000         0         0         0         0         0
         0         0         0         0    9.7434         0         0         0         0
         0         0         0         0         0   37.4903         0         0         0
         0         0         0         0         0         0   79.0165         0         0
         0         0         0         0         0         0         0    1.0000         0
         0         0         0         0         0         0         0         0    1.0000
```

We see that the lower eigenvalues $\lambda_i = 79, 37, 9, 1, 1$ for $i \leq n/2$ are relatively small in comparison to $\lambda_n = 246$. Now we coarsen the mesh using $h = 0.25$ and obtain:

```
A1 =

    1    0    0    0    0
  -16   32  -16    0    0
    0  -16   32  -16    0
    0    0  -16   32  -16
    0    0    0    0    1

V =

   0.00000   0.00000   0.00000   0.69531   0.00000
  -0.50000   0.70711   0.50000   0.56348   0.20461
   0.70711   0.00000   0.70711   0.39644   0.39644
  -0.50000  -0.70711   0.50000   0.20461   0.56348
   0.00000   0.00000   0.00000   0.00000   0.69531

lambda =

   54.6274         0         0         0         0
         0   32.0000         0         0         0
         0         0    9.3726         0         0
         0         0         0    1.0000         0
         0         0         0         0    1.0000
```

We see that the lower eigenvalues $\lambda_i = 1, 1$ for $i < n/2$ are relatively small in comparison to $\lambda_n = 54$. Therefore, a relatively small eigenvalue $\lambda_5 = 79$ in A_2 became a the biggest eigenvalue $\lambda_5 = 54$ on the coarser grid. Now we coarsen again the mesh using $h = 0.5$ and obtain:

```
A0 =

   1   0   0
  -4   8  -4
   0   0   1

V =

   0.00000   0.86824   0.00000
```

```
   1.00000   0.49614   0.49614
   0.00000   0.00000   0.86824

lambda =


   8   0   0
   0   1   0
   0   0   1
```

And again, a small eigenvalue in A_1 became the biggest eigenvalue $\lambda_3 = 8$ on the coarsest mesh. Of course, this behavior of the eigenvalues and eigenvalues is due to the structure of the linear equation system and therefore due to the problem statement; here Poisson.

10.7.5 Variational geometric multigrid

Usually the systematic procedure for multigrid is first explained for a two-grid method. This idea is then extended to *multiple* meshes.

Let $V_h = \{\varphi_1, \ldots, \varphi_n\}$, $dim(V_h) = n$, and we consider the well-known abstract problem: Find $u_h \in V_h$ such that

$$a(u_h, \varphi_h) = l(\varphi_h) \quad \forall \varphi_h \in V_h.$$

We know for $v_h \in V_h$:

$$v_h = \sum_{i=1}^{n} x_i \varphi_i$$

with $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ being the coefficient vector. Consequently, we have the correspondance:

$$x \in \mathbb{R}^n \quad \leftrightarrow \quad v_h \in V_h \tag{225}$$

with the relation $dim(V_h) = n$. We now consider hierarchic refinement and ask that the finite element spaces are nested:

$$V_{2h}^{(1)} \subset V_h^{(1)}$$

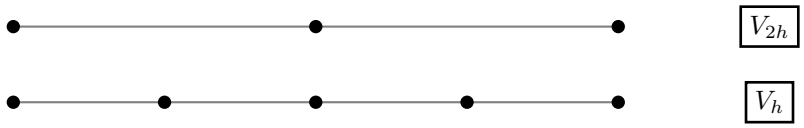


Figure 40: Hierarchical refinement and nexted FEM spaces $V_{2h} \subset V_h$.

Remark 10.31. *Such hierarchical refinements can also be used adjoint-based mesh refinement in Chapter 9.*

Formulation 10.32 (Two-grid method)**.** *A two-grid method can be formulated as:*

1. *Let $u_h^k \in V_h$ be given. Compute $u_h^{k,1} \in V_h$ (on the fine grid $\mathcal{T}_h$) by iterating ν times Richardson (usually only $1 \leq \nu \leq 3$). This means (using (225)):*

$$\begin{aligned} \textit{Set } x_h^{k,0} &= x_h^k \\ \textit{Iterate } x_h^{k,i/\nu} &= x_h^{k,(i-1)/\nu} + \omega(b - A_h x_h^{k,(i-1)/\nu}), \quad i = 1, \ldots, \nu. \end{aligned}$$

   For $i = \nu$, we have obtained $x_h^{k,1} \in \mathbb{R}^{dim(V_h)}$.

2. *Coarse grid correction: we now solve the problem on the coarser grid $\mathcal{T}_{2h}$ with V_{2h}. Find $w \in V_{2h}$ such that*

$$a(u_h^{k,1}|_{\mathcal{T}_{2h}} + w, v) = l(v) \quad \forall v \in V_{2h}, \tag{226}$$

where $u_h^{k,1}|_{\mathcal{T}_{2h}}$ is now restricted to the coarser mesh.

3. *Set*

$$u_h^{k+1} = u_h^{k,1} + w|_{\mathcal{T}_h},$$

where w is now prolongated back to the fine grid.

We explain the second step now in more detail. We define the bases

$$V_h = \{\varphi_1^h, \ldots, \varphi_n^h\}, \qquad \dim(V_h) = n$$
$$V_{2h} = \{\varphi_1^{2h}, \ldots, \varphi_m^{2h}\} \qquad \dim(V_{2h}) = m.$$

Clearly, $m < n$. Since $V_{2h} \subset V_h$, there exist coefficients r_{ij} to restrict the fine grid function to the coarse grid:

$$\varphi_i^{2h} = \sum_{j=1}^{n} r_{ij}\varphi_j^h, \quad 1 \le i \le m$$

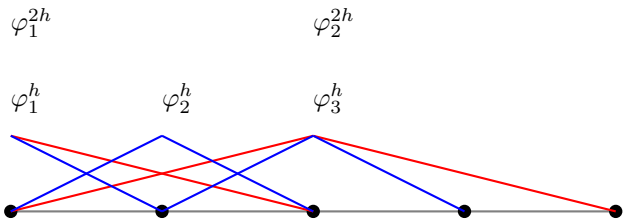


Figure 41: Illustration of $\varphi_i^{2h} = \sum_{j=1}^{n} r_{ij}\varphi_j^h$.

The question is how to compute the restriction operator (matrix) $R = (r_{ij})$? Using the above relation, this can be done explicitly. For instance, observing Figure 41, we see

$$\varphi_1^{2h} = r_{11}\varphi_1^h + r_{12}\varphi_2^h + \ldots + r_{15}\varphi_5^h$$

In more detail, φ_1^{2h} can be constructed as:

$$\varphi_1^{2h} = 1\varphi_1^h + 0.5\varphi_2^h$$

yielding $r_{11} = 1, r_{12} = 0.5, r_{1j} = 0$ for $j = 3, 4, 5$. Extending to φ_2^{2h} and φ_3^{2h}, we obtain:

$$R_{2h}^h = \begin{pmatrix} 1 & 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 1 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 & 1 \end{pmatrix} \quad \in \mathbb{R}^{3\times 5}$$

Remark 10.33. *As usual, in the actual implementation, care has to be taken how the boundary conditions are treated in the FEM code and possibly the indices corresponding to boundary values may be modified.*

We now insert this basis representation into (226):

$$\begin{aligned}
& a(u_h^{k,1}+w,\varphi^{2h}) = l(\varphi^{2h}) && \forall \varphi^{2h}\in V_{2h}\\
\Leftrightarrow\quad & a(w,\varphi^{2h}) = l(\varphi^{2h}) - a(u_h^{k,1},\varphi^{2h}) && \forall \varphi^{2h}\in V_{2h}\\
\Leftrightarrow\quad & a\left(\sum_{j=1}^{m} y_j\varphi_j^{2h},\varphi_i^{2h}\right) = l(\varphi_i^{2h}) - a\left(\sum_{s=1}^{n} x_s^{k,1}\varphi_s^h,\varphi_i^{2h}\right), && i=1,\dots,m\\
\Leftrightarrow\quad & \sum_{j=1}^{m} y_j a\left(\varphi_j^{2h},\varphi_i^{2h}\right) = l\left(\sum_{s=1}^{n} r_{is}\varphi_s^h\right) - a\left(\sum_{s=1}^{n} x_s^{k,1}\varphi_s^h,\sum_{s=1}^{n} r_{is}\varphi_s^h\right)\\
\Leftrightarrow\quad & \sum_{j=1}^{m} y_j a\left(\varphi_j^{2h},\varphi_i^{2h}\right) = \sum_{s=1}^{n} r_{is}\left(l\left(\varphi_s^h\right) - a\left(\sum_{t=1}^{n} x_t^{k,1}\varphi_t^h,\varphi_s^h\right)\right)\\
\Leftrightarrow\quad & A_{2h}y_{2h} = R_{2h}^h(b_h - A_h x_h^{k,1})
\end{aligned}$$

with the dimensions

$$A_{2h}\in\mathbb{R}^{m\times m},\quad y_{2h}\in\mathbb{R}^m,\quad R_{2h}^h\in\mathbb{R}^{m\times n},\quad b_h\in\mathbb{R}^n,\quad A_h\in\mathbb{R}^{n\times n},\quad x_h^{k,1}\in\mathbb{R}^n.$$

Specifically, the rectangular restriction matrix is determined by $(R_{2h}^h)_{ij} = r_{ij}$.

Algorithm 10.34 (Two grid method (TGM) - algebraic version)**.** *We denote the linear equation systems by $A_h x_h = b_h$ and $A_{2h}x_{2h} = b_{2h}$ with $A_h\in\mathbb{R}^{n\times n}$ and $A_{2h}\in\mathbb{R}^{m\times m}$ with $m<n$. Let the current iterate $x_h^k\in\mathbb{R}^n$ be given. The following method computes the next iterate $x_h^{k+1}\in\mathbb{R}^n$:*

$TGM(x_h^k)$
{
$x_h^{k,0} = x_h^k$
for $(i=1,\dots,\nu)$
  $x_h^{k,i/\nu} = x_h^{k,(i-1)/\nu} + \omega(b - A_h x_h^{k,(i-1)/\nu})$ *Pre-smoothing*
$d_h = b_h - A_h x_h^{k,1}$ *Defect on fine grid*
$d_{2h} = R_{2h}^h d_h$ *Restriction to coarse grid*
$x_{2h} = A_{2h}^{-1} d_{2h}$ *Coarse grid solve*
$y_h = (R_{2h}^h)^T x_{2h}$ *Prolongation to fine grid*
$x_h^{k,2} = x_h^{k,1} + y_h$ *Coarse grid correction applied to fine grid solution*
return $x_h^{k,2}$
}

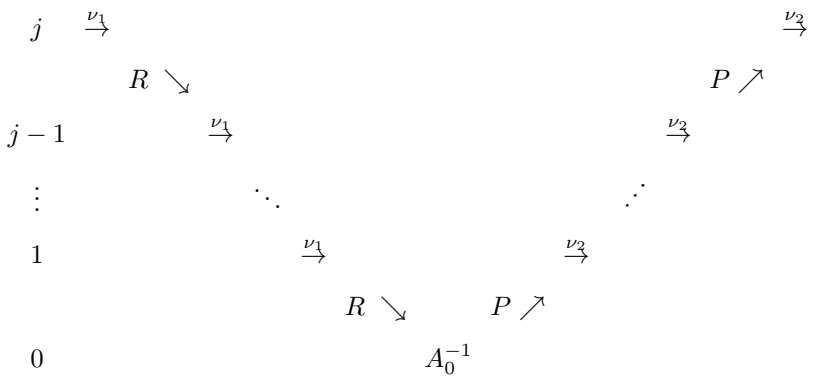
We see in this algorithm, the two-grid method can be extended to a multigrid method by applying recursively the idea to the coarse grid solve $y_{2h} = A_{2h}^{-1}d_{2h}$.

Remark 10.35 (Solve on the coarsest grid)**.** *If A_{2h} is sufficiently small (i.e., $\mathcal{T}_{2h}$ sufficiently coarse), then $x_{2h} = A_{2h}^{-1}d_{2h}$ is often solved in practice with a direct solver, e.g, LU.*

Algorithm 10.36 (Geometric multigrid (GMG) method)**.** *Let $j\in\mathbb{N}$ be the mesh level. For instance $j\hat{=}h$, then $j-1\hat{=}2h$, and so forth. Let $\nu_1,\nu_2\in\mathbb{N}$ be given parameters for the number of Richardson iterations. Moreover, let $\gamma\in\mathbb{N}$ be the cycle form parameter (often $\gamma=1$ or $\gamma=2$). The following method computes the new iterate x_j^{k+1} on mesh level j:*

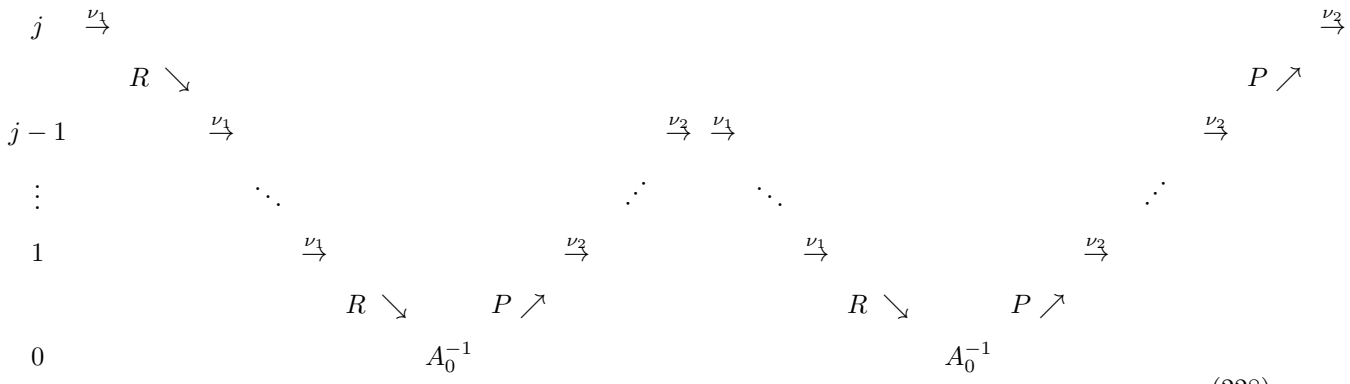
$GMG(j, x_j^k, b_j)$
{
$if\ (j == 0)$
  {
    $x_j^k = A_j^{-1} b_j$ *Solve on coarsest mesh with direct solve for instance*
    return x_j^k
  }
$x_j^{k,0} = x_j^k$
$for\ (i = 1, \ldots, \nu_1)$
  $x_j^{k,i/\nu} = x_j^{k,(i-1)/\nu} + \omega(b_j - A_j x_j^{k,(i-1)/\nu})$ *Pre-smoothing*
$d_j = b_j - A_j x_j^{k,1}$ *Defect on fine grid*
$d_{j-1} = R_{j-1}^j d_j$ *Restriction to coarse grid*
$x_{j-1} = 0$ *Initial value for coarse grid solution*
//Coarse grid soluton now with multigrid
$if\ (j == 1)$
  $\bar{\gamma} = 1$
else
  $\bar{\gamma} = \gamma$
$for\ (i = 1, \ldots, \bar{\gamma})$
  $x_{j-1} = GMG(j-1, x_{j-1}, d_{j-1})$
$y_j = (R_{j-1}^j)^T x_{j-1}$ *Prolongation to fine grid*
$x_j^{k,2} = x_j^{k,1} + y_j$ *Coarse grid correction applied to fine grid solution*
$for\ (i = 1, \ldots, \nu_2)$
  $x_j^{k,2+i/\nu} = x_j^{k,2+(i-1)/\nu} + \omega(b_j - A_j x_j^{k,2+(i-1)/\nu})$ *Post-smoothing*
return $x_j^{k,3}$
}

The choice of γ gives the GMG method specific names. For instance $\gamma = 1$ results in a so-called V cycle. The choice $\gamma = 2$ is known as W cycle. Here, we do one V cycle from j to 0 and back to $j-1$ and do a second V cycle from $j-1$ to 0 back to j.



(227)

The V cycle.



(228)

The W cycle.

10.7.6 Usage of MG in practice and MG as a preconditioner

In practice, multigrid solvers can be used as solvers for $Ax = b$ themselves. Or, which happens more often, multigrid methods are used a preconditioner (Section 10.5) for other iterative methods, e.g., CG or GMRES. **Advantages** are twofold:

- There is no need to solve MG up to a tolerance;
- In a giving programming code with some good basic solver only the preconditioner needs to be replaced.

The performance of MG as a preconditioner is shown in Section 10.8 in which CG is used as linear solver, preconditioned with MG and SOR (successive overrelaxation) as smoother.

10.7.7 Convergence analysis of the W cycle

For full proofs, we refer to [13, 24, 67]. Still following [13], we concentrate on the W cycle because the proofs are simpler. The proofs of the V cycle are more involved requiring sufficient smoothing steps and H^2 regularity.

Let j be the mesh index and k the iteration index. The goal is to derive an estimation with the multigrid convergence rate ρ_j on $j+1$ meshes:

$$\|u^{j,k+1} - u_j\| \leq \rho_j \|u^{j,k} - u_j\|$$

where $u_j \in V_j$ is the discrete solution on mesh level j and $u^{j,k+1}$ is the $k+1$-th iterate. Specifically, **the multigrid method is convergent when $\rho_j < 1$ and ρ_j is independent of the mesh size h** (see Section 10.7.3 for 1D Poisson and Richardson as a smoother). Then, the computational cost would not increase while refining the mesh. The factor ρ_j is the convergence rate. Of course, $\rho_j \ll 1$, the faster the convergence.

Error recursion for the smoothing step

We have:

$$e^{k,1} = x_h - x_h^{k,1} = (I_h - \omega A_h)^{\nu_1}(x_h - x_h^k) = S_h^{\nu_1} e^k$$

Coarse grid correction step

Then:

$$\begin{aligned} e^{k+1} &= x_h - x_h^{k+1} \\ &= x_h - \left(x_h^{k,1} + (R_{2h}^h)^T A_{2h}^{-1} R_{2h}^h (b_h - A_h x_h^{k,1}) \right) \\ &= e^{k,1} - (R_{2h}^h)^T A_{2h}^{-1} R_{2h}^h A_h e^{k,1} \\ &= (I_h - (R_{2h}^h)^T A_{2h}^{-1} R_{2h}^h A_h) e^{k,1} \end{aligned}$$

With this, we obtain for the iteration matrix:

$$e^{k+1} = (I_h - (R_{2h}^h)^T A_{2h}^{-1} R_{2h}^h A_h) S_h^{\nu_1} e^k = (A_h^{-1} - (R_{2h}^h)^T A_{2h}^{-1} R_{2h}^h) A_h S_h^{\nu_1} e^k$$

This allows us to split the error into two parts:

$$\|e^{k+1}\| \leq \underbrace{\|(A_h^{-1} - (R_{2h}^h)^T A_{2h}^{-1} R_{2h}^h)\|}_{\text{Approximation}} \underbrace{\|A_h S_h^{\nu_1}\|}_{\text{Smoothing}} \|e^k\|.$$

The appropriate norm is the Euclidian norm as it will turn out.

10.7.7.1 Norms We recall that A is s.p.d. (only necessary for $s = 1$ in the following). Thus we define (similar to the CG method):

$$\|x\|_s := \sqrt{(x, A^s x)}, \quad s = 0, 1, 2.$$

In detail:

$$\begin{aligned} s = 0: &\quad \|x\|_0 = \sqrt{((x,x))} \quad \text{Euclidian norm} \\ s = 1: &\quad \|x\|_1 = \sqrt{((x,Ax))} \quad \text{Energy norm} \\ s = 2: &\quad \|x\|_2 = \sqrt{((Ax,Ax))} \quad \text{Defect norm} \end{aligned}$$

These definitions can be further extended to $s \in \mathbb{R}$ by using a diagonalization of A.

The Sobolev norms $\|\cdot\|_{L^2}$ and $\|\cdot\|_{H^1}$ can be related to $\|\cdot\|_0$ and $\|\cdot\|_1$, respecively. We have:

$$\|x\|_1^2 = (x, Ax) = a(u_h, u_h)$$

As in the Lax-Milgram lemma, cf. Section 8.10, we obtain:

$$\alpha \|u_h\|_{H^1} \leq \|x\|_1^2 = a(u_h, u_h) \leq \gamma \|u_h\|_{H^1}.$$

As seen in the stability estimate of the proof of the Lax-Milgram lemma, these estimates are independent of the basis of V_h and only use approximation properties.

Lemma 10.37. *Let $\{\varphi_1, \ldots, \varphi_N\}$ be the Lagrange basis of V_h on a family of uniform and shape regular triangulations. Let x be the coefficient vector of $v_h \in V_h$. Then, there exist $c_1, c_2 > 0$ independent of h, but dependent on the overall mesh $\mathcal{T}_h$ such that*

$$c_1 h^{n/2} \|x\|_0 \leq \|v\|_{L^2} \leq c_2 h^{n/2} \|x\|_0.$$

Here n is the space dimension.

10.7.7.2 Approximation properties

Lemma 10.38 (Approximation property)**.** *Let $A_h x_h = b_h$ the discrete problem on the finest grid. Moreover, let $x_h^{k,1}$ denote the iterate after smoothing and $x_h^{k,2}$ denotes the iterate after the coarse grid correction. If the mesh is uniform and shape-regular, the variational problem is H^2 regular and we have*

$$\|x_h - x_h^{k,2}\|_0 \leq c h^{2-n} \|x_h - x_h^{k,1}\|_2$$

As before, n is the space dimension. We see that this estimate is not robust w.r.t. to n.

10.7.7.3 Smoothing properties

Lemma 10.39 (Smoothing property)**.** *Let A_h be s.p.d. The Richardson iteration*

$$x_h^{k+1} = x_h^k + \omega(b_h - A_h x_h^k)$$

with $\omega = (\lambda_{max}(A))^{-1}$ yields the following estimate:

$$\|x_h - x_h^\nu\|_2 \leq \frac{\lambda_{max}(A_h)}{\nu} \|x_h - x_h^0\|_0.$$

10.7.7.4 Two- and multigrid convergence results

Theorem 10.40. *With all assumptions from before, the two-grid method with ν steps of Richardson iterations as smoother, satisfies*

$$\|x_h - x_h^{k+1}\|_0 \leq \frac{c}{\nu}\|x_h - x_h^0\|_0$$

To this end, for sufficiently large ν, the two-grid method converges independendly of the mesh size h.

Lemma 10.41 (Recursion; [13] or [24][Chapter V.3)**.** *Let ρ_1 be the convergence rate of the two-grid method and ρ_l the convergence rate of a multigrid method with the cycle parameter γ and $l \geq 2$ levels. Then, the following recursion holds true*

$$\rho_l \leq \rho_1 + (1 + \rho_1)\rho_{l-1}^{\gamma}$$

Theorem 10.42 (Convergence ρ_l rate W cycle)**.** *If $\rho_1 \leq \frac{1}{5}$ for the two-grid method, then it holds for the W cycle of the multigrid method*

$$\rho_l \leq \frac{5}{3}\rho_1 \leq \frac{1}{3} \quad \text{for } l = 2, 3, \ldots$$

Lemma 10.43 (Optimal complexity of the multigrid method; [13])**.** *Let N_l the number of unknown on level l using P_1 finite elements. Then, the amount of arithemetic operations A_l on level l is*

$$A_l = O(N_l).$$

10.8 Numerical tests

We continue the numerical tests from Section 8.16 and consider again the Poisson problem in 2D and 3D on the unit square and with force $f = 1$ and homogeneous Dirichlet conditions. We use as solvers:

- CG
- PCG with SSOR preconditioning and $\omega = 1.2$
- PCG with geometric multigrid preconditioning and SOR smoother
- GMRES
- GMRES with SSOR preconditioning and $\omega = 1.2$
- BiCGStab
- BiCGStab with SSOR preconditioning and $\omega = 1.2$

The tolerance is chosen as $TOL = 1.0e - 12$. We also run on different mesh levels in order to show the dependency on n.

Dimension	Elements	DoFs(N)	CG	PCG	PCG-GMG	GMRES	P-GMRES	BiCGStab	P-BiCGStab
2	16	25	3		5				
2	64	81	10		6				
2	256	289	23	19	6	23	18	16	12
2	1024	1089	47	33	6	83	35	33	21
2	4096	4225	94	60	6	420	78	66	44
2	16384	16641	186		6				
2	65536	66049	370		6				
2	262144	263169	731		5				
2	1048576	1050625	1400		5				
2	4194304	4198401	2780		5				
3	4096	4913	25	19		25	21	16	11
3	32768	35937	51	32		77	38	40	23
3	262144	274625	98	57		307	83	69	46

Four main messages:

- GMRES performs worst.
- PCG performs better than pure CG, but less significant than one would wish.
- BiCGStab performs very well - a bit surprising.
- Geometric multigrid preconditioning yields mesh-independent constant iteration numbers of the outer CG iteration

10.9 Chapter summary and outlook

In extension to Numerik 1, we introduced in this chapter, the numerical solution with geometric multigrid methods. As smoother, however, iterative methods from Numerik 1, are needed. This shows a nice combinations between different classes in numerical methods.

11 Applications in continuum mechanics: linearized elasticity and Stokes

In this chapter, we undertake a short excursus to important prototype applications: linearized elasticity (solid mechanics) and the Stokes problem (fluid mechanics). Both models lead to systems of equations since they are vector-valued, but are still linear and consequently fit into the lines of our previous developments. We try to provide an introduction that touches the aspects, which can be understood with the help of our previous developments. For instance, linearized elasticity is still of elliptic type and we show, how the well-posedness can be proven using Lax-Milgram (i.e., Riesz since the problem is symmetric). On the other hand, the maximum principle does not hold anymore. These notes cannot provide all possible details since they fill entire books such as [34] and [125]. In regard of fluid mechanics, the Navier-Stokes equations have been introduced in Section 9.24 to demonstrate mesh adaptivity with respect to drag forces as goal functional.

11.1 Modeling

Since linearized elasticity is vector-valued, we design now the space H^1 for three solution components $\hat{u}_x, \hat{u}_y, \hat{u}_z$:

$$\hat{u} = (\hat{u}_x, \hat{u}_y, \hat{u}_z) \in [H^1(\widehat{\Omega})]^3$$

Specifically, we define:

$$\hat{V}_s^0 := [H_0^1(\widehat{\Omega})]^3.$$

Remark 11.1. *In linearized elasticity we deal with small displacements and consequently the two principle coordinate systems, Lagrangian and Eulerian are identified; we also refer the reader to Remark 4.14. For large displacements, one needs to distingish between a* ***reference configuration*** *in which the actual computations are carried out. This reference configuration is indicated in a 'hat' notation, i.e., $\widehat{\Omega}$. Via a transformation, the solution is mapped to the* ***current (i.e., physical or deformed) configuration*** *Ω.*

Problem 11.2 (Stationary linearized elasticity)**.** *Let $\Omega \subset \mathbb{R}^d, d = 3$. Given $\hat{f} : \Omega \to \mathbb{R}^d$, find a vector-valued displacement $\hat{u} \in \hat{V}_s^0 = H_0^1$ such that*

$$(\widehat{\Sigma}_{lin}, \widehat{\nabla}\hat{\varphi}) = (\hat{f}, \hat{\varphi}) \quad \forall \hat{\varphi} \in \hat{V}_s^0,$$

where

$$\widehat{\Sigma}_{lin} = 2\mu\widehat{E}_{lin} + \lambda tr\widehat{E}_{lin}\hat{I}.$$

Here, $\widehat{\Sigma}_{lin}$ is a matrix and specified below. Furthermore, the gradient $\widehat{\nabla}\hat{\varphi}$ is a matrix. Consequently, the scalar product is based on the Frobenius Definition 8.14. The material parameters $\mu, \lambda > 0$ are the so-called Lamé parameters. Finally $tr(\cdot)$ is the trace operator. For further details on the entire model, we refer to Ciarlet [34].

Definition 11.3 (Green-Lagrange strain tensor $\widehat{E}$)**.**

$$\widehat{E} = \frac{1}{2}(\widehat{C} - \hat{I}) = \frac{1}{2}(\widehat{F}^T\widehat{F} - \hat{I}) = \frac{1}{2}(\widehat{\nabla}\hat{u} + \widehat{\nabla}\hat{u}^T + \widehat{\nabla}\hat{u} \cdot \widehat{\nabla}\hat{u}^T),$$

which is again symmetric and positive definite for all $\hat{x} \in \widehat{\Omega}$ since $\widehat{C}$ and of course, $\hat{I}$ have these properties. $\diamond$

Performing geometric linearization, which is reasonable for example when $\|\widehat{\nabla}\hat{u}\| \ll 1$, we can work with the linearized Green-Lagrange strain tensor

Definition 11.4 (Linearized Green-Lagrange strain tensor $\widehat{E}_{lin}$)**.**

$$\widehat{E}_{lin} = \frac{1}{2}(\widehat{\nabla}\hat{u} + \widehat{\nabla}\hat{u}^T).$$

$\diamond$

Proposition 11.5. *Let the following linear boundary value problem be given:*

$$-div(\widehat{\Sigma}_{lin}) = f \quad \text{in } \widehat{\Omega}, \tag{229}$$

$$\hat{u} = 0 \quad \text{on } \partial\widehat{\Omega}_D, \tag{230}$$

$$\widehat{\Sigma}_{lin}\hat{n} = \hat{g} \quad \text{on } \partial\widehat{\Omega}_N. \tag{231}$$

Finding a solution $\hat{u}$ of this strong form is formally equivalent to finding a solution of Problem 11.2.

Proof. We employ Green's formula for any sufficiently smooth tensor field $\widehat{\Sigma}$ and vector field $\hat{\varphi}$, we obtain

$$\int_{\widehat{\Omega}} \widehat{\nabla} \cdot \widehat{\Sigma} \cdot \hat{\varphi}\, d\hat{x} = -\int_{\widehat{\Omega}} \widehat{\Sigma} : \widehat{\nabla}\hat{v}\, d\hat{x} + \int_{\partial\widehat{\Omega}_N} \widehat{\Sigma}\hat{n} \cdot \hat{\varphi}\, d\hat{s} = -\int_{\widehat{\Omega}} \widehat{\Sigma} : \widehat{E}_{lin}\, d\hat{x} + \int_{\partial\widehat{\Omega}_N} \widehat{\Sigma}\hat{n} \cdot \hat{\varphi}\, d\hat{s}.$$

The last equal sign is justified with linear algebra arguments, i.e., for a symmetric matrix A is holds:

Definition 11.6 (Proof is homework)**.** *For the Frobenius scalar product, and $A = A^T$, it holds*

$$A : B = A : \frac{1}{2}(B + B^T).$$

Furthermore, on the boundary part $\partial\widehat{\Omega}_D$ the vector field $\hat{\varphi}$ vanishes by definition. Thus, the first direction is shown. Conversely, we now assume that the variational equations are satisfied; namely,

$$\int_{\widehat{\Omega}} \widehat{\Sigma} : \widehat{\nabla}\hat{\varphi}\, d\hat{x} = \int_{\widehat{\Omega}} \hat{f} \cdot \hat{\varphi}\, d\hat{x}$$

if $\hat{\varphi} = 0$ on all $\partial\Omega$. By Green's formula we now obtain

$$\int_{\widehat{\Omega}} \widehat{\Sigma} : \widehat{\nabla}\hat{\varphi}\, d\hat{x} = -\int_{\widehat{\Omega}} \widehat{\nabla} \cdot \widehat{\Sigma}\, d\hat{x}.$$

Putting both pieces together and taking into account that the integrals do hold on arbitrary volumes yields

$$-\widehat{\nabla} \cdot \widehat{\Sigma} = \hat{f} \quad \text{in } \widehat{\Omega}.$$

Considering now the Neumann boundary, we use again Green's formula to see

$$\int_{\partial\widehat{\Omega}_N} \widehat{\Sigma}\hat{n} \cdot \hat{\varphi}\, ds = \int_{\partial\widehat{\Omega}_N} \hat{g}\, ds.$$

This holds for arbitrary boundary parts $\partial\widehat{\Omega}_N$ such that $\widehat{\Sigma}\hat{n} = \hat{g}$ can be inferred and concluded the second part of the proof. □

11.2 Well-posedness

Even so that linearized elasticity is an elliptic problem and has much in common with the scalar-valued Poisson problem, establishing existence and uniquenss for the stationary version is a non-trivial task. Why? Let us work with Problem 11.2 in 3D and since this problem is linear we aim to apply the Riesz representation theorem[9] (or the Lax-Milgram lemma - see Theorem 8.72) and need to check the assumptions:

- Determine in which space $\hat{V}$ we want to work;
- The form $A(\hat{u}, \hat{\varphi})$ is symmetric or non-symmetric
- The form $A(\hat{u}, \hat{\varphi})$ is continuous bilinear w.r.t. to the norm of $\hat{V}$;
- The bilinear form $A(\hat{u}, \hat{\varphi})$ is V-elliptic;

[9] Here, $A(\hat{u}, \hat{\varphi}) = (\widehat{\Sigma}_{lin}, \widehat{\nabla}\hat{\varphi})$ is symmetric and the Riesz representation 8.75 is sufficient for existence and uniqeness!

- The right-hand side functional $\hat{f}$ is a continuous linear form.

The ingredients to check are:

- From which space must $\hat{f}$ be chosen such that it is continuous? (This leads to Sobolev embedding theorems [34, 50]);
- How do we check the V-ellipticity? (This requires the famous Korn inequality as it is stated in a minute below).

For proofing existence of linearized elasticity, we need to check as second contition the V-ellipticity which is 'easy for scalar-valued problems' but non-trivial in the case of vector-valued elasticity. The resulting inequalities are named after Korn (1906):

Theorem 11.7 (1st Korn inequality)**.** *Let us assume displacement Dirichlet conditions on the entire boundary $\partial\Omega$. For vector-fields $\hat{u} \in H_0^1(\widehat{\Omega})^3$ it holds:*

$$\|\nabla \hat{u}\| \leq c_{Korn} \|\widehat{E}_{lin}(\hat{u})\|.$$

where $\widehat{E}_{lin}(\hat{u}) := \frac{1}{2}(\widehat{\nabla}\hat{u}_s + \widehat{\nabla}\hat{u}_s^T) \in L^2(\widehat{\Omega}_s)$.

Proof. See for example [24, 34, 102]. □

The more general form is as follows:

Theorem 11.8 (2nd Korn inequality)**.** *Let a part of the boundary of Neumann type, i.e., $\partial\Omega_N \neq 0$ and let the entire boundary be of class C^1, or a polygon, or of Lipschitz-type. For vector-fields $\hat{u} \in H^1(\widehat{\Omega})^3$ and $\widehat{E}_{lin} \in L^2$ it holds*

$$\|\hat{u}_s\|_{H^1(\widehat{\Omega}_s)} \leq C_{Korn}\big(|\hat{u}_s|^2_{L^2(\widehat{\Omega}_s)} + |\widehat{E}_{lin}|^2_{L^2(\widehat{\Omega}_s)}\big)^{1/2} \quad \forall \hat{u}_s \in H^1(\widehat{\Omega}_s),$$

and a positive constant C_K and secondly,

$$u \mapsto \big(|\hat{u}_s|^2_{L^2(\widehat{\Omega}_s)} + |\widehat{E}_{lin}|^2_{L^2(\widehat{\Omega}_s)}\big)^{1/2}$$

is a norm that is equivalent to $\|\cdot\|_{H^1}$.

Theorem 11.9 (Existence of a weak solution of linearized elasticity)**.** *Let $\widehat{\Omega} \subset \mathbb{R}^3$ and $\widehat{\Gamma}_D > 0$ and $\widehat{\Gamma}_N > 0$ Dirichlet and Neumann boundary respectively. Let the Lamé constants be $\mu > 0$ and $\lambda > 0$ and $\hat{f} \in L^{6/5}(\Omega)$ and $\hat{g} \in L^{4/3}(\widehat{\Gamma}_N)$ be given. Furthermore, let*

$$\hat{V} := \{\hat{\varphi} \in H^1(\widehat{\Omega}) |\, \hat{\varphi} = 0 \text{ on } \widehat{\Gamma}_D\}.$$

Then, there exists a unique element $\hat{u} \in \hat{V}$ such that

$$A(\hat{u}, \hat{\varphi}) = F(\hat{\varphi}) \quad \forall \hat{\varphi} \in \hat{V}.$$

Here, the bilinear form and the right hand side functional are given by

$$A(\hat{u}, \hat{\varphi}) = (\widehat{\Sigma}_{lin}, \widehat{\nabla}\hat{\varphi}), \quad \text{and} \quad F(\hat{\varphi}) = (\hat{f}, \hat{\varphi}) + \langle \hat{g}, \hat{\varphi} \rangle, \tag{232}$$

where the linearized STVK model is given by

$$\widehat{\Sigma}_{lin} := 2\mu \widehat{E}_{lin} + \lambda tr\widehat{E}_{lin}\hat{I}.$$

The function $\hat{g}$ is a prescribed traction condition on $\widehat{\Gamma}_N$; namely:

$$[2\mu \widehat{E}_{lin} + \lambda tr\widehat{E}_{lin} I]\, \hat{n} = \hat{g}.$$

Proof. The proof is as follows:

- **Continuity of the right hand side functional $F(\cdot)$.** In order to apply Hölder's inequality, we first check the correct Sobolev embedding:

$$H^1 = W^{1,2} \hookrightarrow L^{p^*} \quad \text{with } \frac{1}{p^*} = \frac{1}{p} - \frac{m}{d}$$

for $m < \frac{d}{p}$. Since we work in 3D, $d = 3$. Furthermore, $m = 1, p = 2$ in $W^{m,p}$. It can be inferred that consequently, $p^* = 6$. For the continuity of F, we calculate with Hölder's inequality:

$$|(\hat{f}, \hat{\varphi})| \leq \|\hat{f}\|_p \, \|\hat{\varphi}\|_{q=6}$$

where $\frac{1}{p} + \frac{1}{q} = 1$. Since, $q = 6$, this yields $p = \frac{6}{5}$, i.e., $\hat{f} \in L^{6/5}$. In the same spirit, we obtain $\hat{g} \in L^{4/3}$ while additionally employing the trace inequality. Let us verify this claim: From trace theorems, e.g., [34], p. 280, we know that for $1 \leq p < \infty$ we have

$$tr \in L(W^{1,p}, L^{p^*}) \quad \Leftrightarrow \quad \|\cdot\|_{L^{p^*}} \leq c\|\cdot\|_{W^{1,p}},$$

with $\frac{1}{p^*} = \frac{1}{p} - \frac{1}{(d-1)}(\frac{p-1}{p})$ if $1 \leq p < d$. As before and still, $d = 3$. This means $1 \leq p \leq 2$. Of course we want to work again with $W^{1,2}$, $p = 2$. Then,

$$\frac{1}{p^*} = \frac{1}{p} - \frac{1}{(d-1)}(\frac{p-1}{p}) = \frac{1}{2} - \frac{1}{(3-1)}(\frac{2-1}{2}) = \frac{1}{4}.$$

Consequently, $p^* = 4$. Let us estimate with Hölder's inequality and the trace theorem:

$$\int_{\widehat{\Gamma}_N} \hat{g}\hat{\varphi} \, ds \leq \|\hat{g}\|_{L^q(\widehat{\Gamma}_N)} \|\hat{\varphi}\|_{L^4(\widehat{\Gamma}_N)} \leq \|\hat{g}\|_{L^q(\widehat{\Gamma}_N)} \|\hat{\varphi}\|_{W^{1,2}(\widehat{\Omega})}.$$

In order to be able to apply Hölder's inequality we need to find the correct q. This is now easy with

$$\frac{1}{p} + \frac{1}{q} = 1 \quad \Rightarrow q = \frac{4}{3}.$$

Consequently, $\hat{g} \in L^{4/3}$.

- **Symmetry of $A(\hat{u}, \hat{\varphi})$.** Let us now verify if $A(\hat{u}, \hat{\varphi})$ is symmetric:

$$\begin{aligned}
A(\hat{u}, \hat{\varphi}) &= (\widehat{\Sigma}_{lin}, \widehat{\nabla}\hat{\varphi}) = (2\mu\widehat{E}_{lin}(\hat{u}) + \lambda tr(\widehat{E}_{lin}(\hat{u}))\hat{I}, \widehat{E}_{lin}(\hat{\varphi})) \\
&= 2\mu(\widehat{E}_{lin}(\hat{u}), \widehat{E}_{lin}(\hat{\varphi})) + \lambda(tr(\widehat{E}_{lin}(\hat{u}))\hat{I}, \widehat{E}_{lin}(\hat{\varphi})) \\
&= 2\mu(\widehat{E}_{lin}(\hat{\varphi}), \widehat{E}_{lin}(\hat{u})) + \lambda(tr(\widehat{E}_{lin}(\hat{\varphi}))\hat{I}, \widehat{E}_{lin}(\hat{u})) \\
&= (2\mu\widehat{E}_{lin}(\hat{\varphi}) + \lambda tr(\widehat{E}_{lin}(\hat{\varphi}))\hat{I}, \widehat{E}_{lin}(\hat{u})) \\
&= A(\hat{\varphi}, \hat{u}).
\end{aligned}$$

Here, we used the relation defined in Definition 11.6.

- **Continuity of $A(\hat{u}, \hat{\varphi})$.** We need to show that

$$|A(\hat{u}, \hat{\varphi})| \leq c\|\hat{u}\|_{H^1}\|\hat{\varphi}\|_{H^1} \quad \forall \hat{u}, \phi \text{ in } \hat{V}.$$

Let us start with the definition and apply first Cauchy's inequality:

$$\begin{aligned}
|A(\hat{u}, \hat{\varphi})| &= |(\widehat{\Sigma}_{lin}, \widehat{\nabla}\hat{\varphi})| = |(\widehat{\Sigma}_{lin}, \widehat{E}_{lin}(\hat{\varphi}))| \\
&\leq \Big(\int |\widehat{\Sigma}_{lin}|^2\Big)^{1/2} \Big(\int |\widehat{E}_{lin}(\hat{\varphi})|^2\Big)^{1/2} \\
&= \Big(\int |2\mu\widehat{E}_{lin}(\hat{u}) + \lambda tr(\widehat{E}_{lin}(\hat{u}))\hat{I}|^2\Big)^{1/2} \Big(\int |\widehat{E}_{lin}(\hat{\varphi})|^2\Big)^{1/2}.
\end{aligned}$$

We use $\|tr(\widehat{E}_{lin}(\hat{u}))\hat{I}\| \leq \|E_{lin}(\hat{u})\|$ and $|E(\hat{u})|_{L^2} \leq c\|\hat{u}\|_{H^1}$, where the latter one specifically holds for all $\hat{u} \in H^1$. Then,

$$\begin{aligned}
|A(\hat{u},\hat{\varphi})| &\leq \Big(\int |2\mu\widehat{E}_{lin}(\hat{u}) + \lambda tr(\widehat{E}_{lin}(\hat{u}))\hat{I}|^2\Big)^{1/2}\Big(\int |\widehat{E}_{lin}(\hat{\varphi})|^2\Big)^{1/2} \\
&\leq \Big(\int |2\mu\widehat{E}_{lin}(\hat{u}) + \lambda\widehat{E}_{lin}(\hat{u})|^2\Big)^{1/2}\Big(\int |\widehat{E}_{lin}(\hat{\varphi})|^2\Big)^{1/2} \\
&\leq c(\mu,\lambda)\Big(\int |\widehat{E}_{lin}|^2\Big)^{1/2}\Big(\int |\widehat{E}_{lin}(\hat{\varphi})|^2\Big)^{1/2} \\
&= c(\mu,\lambda)|E(\hat{u})|_{L^2}\,|E(\hat{\varphi})|_{L^2} \\
&\leq c_1\|\hat{u}\|_{H^1}\,\|\hat{\varphi}\|_{H^1}.
\end{aligned}$$

- **V-ellipticity.** First, we observe that

$$A(\hat{u},\hat{u}) = (2\mu\widehat{E}_{lin}(\hat{u}) + \lambda tr(\widehat{E}_{lin}(\hat{u}))\hat{I}, \widehat{E}_{lin}(\hat{u})) \geq (2\mu\widehat{E}_{lin}(\hat{u}), \widehat{E}_{lin}(\hat{u})) = 2\mu|\widehat{E}_{lin}|_{L^2}.$$

  because $\mu, \lambda > 0$. The V-ellipticity can be inferred if we can show that on the space $\hat{V}_s^0$, the semi-norm

$$\hat{u} \mapsto |\widehat{E}_{lin}|_{L^2}$$

  is a norm that is equivalant to $\|\cdot\|_{H^1}$. To do so, we proceed in two steps. We now employ the 2nd Korn inequality (first step), see Theorem 11.8. In detail: if $\widehat{\Gamma}_D > 0$ and $c > 0$ a given constant such that (the second step)

$$c^{-1}\|\hat{u}\|_1 \leq |\widehat{E}_{lin}|_{L^2} \leq c\|\hat{u}\|_1 \quad \forall \hat{u} \in H^1(\widehat{\Omega}_s),$$

  i.e., on $\hat{V}$, the mapping $u \mapsto |\widehat{E}_{lin}|$ is norm equivalent to $\|\hat{u}\|_1$. (This techniquality requires a seperate proof!)

- **Apply Riesz.** Having proven that A is symmetric, continuous and V-elliptic as well as that F is linear continuous, we have checked all assumptions of the Riesz representation Theorem 8.75 and consequently, a unique solution $\hat{u} \in \hat{V}$ does exist.

For further discussions, the reader might consult Ciarlet [34]. □

Remark 11.10. *Since this elasticity problem is symmetric, the problem can be also interpreted from the energy minimization standpoint as outlined in Theorem 8.75. In this respect, the weak form considered here corresponds to the first-order necessary condition, i.e., the weak form of the Euler-Lagrange equations.*

Remark 11.11. *For linearized elasticity, even that it is of elliptic type, the maximum principle does not hold anymore.*

11.3 Finite element discretization

Problem 11.12 (Discretized stationary linearized elasticity)**.** *Find vector-valued displacements $\hat{u}_h \in \hat{V}_h^0$ such that*

$$(\widehat{\Sigma}_{h,lin}, \widehat{\nabla}\hat{\varphi}_h) = (\hat{f}, \hat{\varphi}_h) \quad \forall \hat{\varphi}_h \in \hat{V}_h^0,$$

where

$$\widehat{\Sigma}_{h,lin} = 2\mu\widehat{E}_{h,lin} + \lambda tr\widehat{E}_{h,lin}\hat{I}.$$

and

$$\widehat{E}_{h,lin} = \frac{1}{2}(\widehat{\nabla}\hat{u}_h + \widehat{\nabla}\hat{u}_h^T).$$

As previsously discussed, the spatial discretization is based upon a space $\hat{V}_h^0$ with basis $\{\hat{\varphi}_1, \ldots, \hat{\varphi}_N\}$ where $N = dim(\hat{V}_h^0)$. We recall, that in 3D elasticity, we have three solution components such that the total dimension is $3N = [\hat{V}_h^0]^d, d = 3$.

We solve the problem:

Formulation 11.13. *Find $\hat{u}_h \in \hat{V}_h$ such that*

$$A(\hat{u}_h, \hat{\varphi}_h) = F(\hat{\varphi}_h) \quad \forall \hat{\varphi}_h \in \hat{V}_h,$$

with

$$A(\hat{u}_h, \hat{\varphi}_h) := (\widehat{\Sigma}_{h,lin}, \widehat{\nabla}\hat{\varphi}_h), \quad F(\hat{\varphi}_h) := (\hat{f}, \hat{\varphi}_h).$$

This relation does in particular hold true for each test function $\hat{\varphi}_i, i = 1, \dots, N$:

$$A(\hat{u}_h, \hat{\varphi}_h^i) = F(\hat{\varphi}_h^i) \quad \forall \hat{\varphi}_h^i, i = 1, \dots, N$$

The solution $\hat{u}_h$ we are seeking for is a linear combination of all test functions, i.e., $\hat{u}_h = \sum_{j=1}^N u_j \hat{\varphi}_h^j$. Inserting this relation into the bilinear form $A(\cdot, \cdot)$ yields

$$\sum_{j=1}^{N} A(\hat{\varphi}_h^j, \hat{\varphi}_h^i) u_j = F(\hat{\varphi}_h^i), \quad i = 1, \dots, N.$$

It follows for the ingredients of the linear equation system:

$$\hat{u} = (\hat{u}_j)_{j=1}^N \in \mathbb{R}^N, \quad b = (F_i)_{i=1}^N \in \mathbb{R}^N, \quad B = A_{ij} = a(\hat{\varphi}_h^J, \hat{\varphi}_h^i).$$

The resulting linear equation systems reads:

$$Bu = b$$

Remark 11.14. *In the matrix B the rule is always as follows: the $\hat{\varphi}_h^i$ test function determines the row and the trial function $\hat{\varphi}_h^j$ the column. This does not play a role for symmetric problems (e.g., Poisson's problem) but becomes important for nonsymmetric problems such as for example Navier-Stokes (because of the convection term).*

Remark 11.15. *In the matrix, the degrees of freedom that belong to Dirichlet conditions (here only displacements since we assume Neumann conditions for the phase-field) are strongly enforced by replacing the corresponding rows and columns as usual in a finite element code.*

Example 11.16 (Laplacian in 3D)**.** *In this vector-valued problem, we have $3N$ test functions since the solution vector is an element of $\mathbb{R}^3$: $u_h = (u_h^{(1)}, u_h^{(2)}, u_h^{(3)})$. Thus in the boundary value problem*

$$\text{Find } u_h \in V_h: \quad (\nabla u_h, \nabla \varphi_h) = (f, \varphi_h) \quad \forall \varphi_h \in V_h,$$

the bilinear form is tensor-valued:

$$a_{Laplacian}(\varphi_h^j, \varphi_h^i) = \int_\Omega \nabla \varphi_h^j : \nabla \varphi_h^i \, dx = \int_\Omega \begin{pmatrix} \partial_1 \varphi_h^{1,j} & \partial_2 \varphi_h^{1,j} & \partial_3 \varphi_h^{1,j} \\ \partial_1 \varphi_h^{2,j} & \partial_2 \varphi_h^{2,j} & \partial_3 \varphi_h^{2,j} \\ \partial_1 \varphi_h^{3,j} & \partial_2 \varphi_h^{3,j} & \partial_3 \varphi_h^{3,j} \end{pmatrix} : \begin{pmatrix} \partial_1 \varphi_h^{1,i} & \partial_2 \varphi_h^{1,i} & \partial_3 \varphi_h^{1,i} \\ \partial_1 \varphi_h^{2,i} & \partial_2 \varphi_h^{2,i} & \partial_3 \varphi_h^{2,i} \\ \partial_1 \varphi_h^{3,i} & \partial_2 \varphi_h^{3,i} & \partial_3 \varphi_h^{3,i} \end{pmatrix} dx.$$

11.4 Numerical tests

11.4.1 3D

We use the above laws given Problem 11.12 and implement them again in deal.II [6, 9].

11.4.1.1 Configuration The domain Ω is spanned by $(0, 0, 0)$ and $(5, 0.5, 2)$ resulting in an elastic beam (see Figure 42). The vertical axis is the y-direction (see again Figure 42). The initial domain is four times globally refined resulting in 4096 and total 14739 DoFs (i.e., 4913 DoFs in each solution component).

11.4.1.2 Boundary conditions The specimen has six boundary planes. On the boundary $\Gamma = \{(x, y, z)|x = 0\}$ the elastic beam is fixed and can move at the other boundaries:

$$\begin{aligned} u = (u_x, u_y, u_z) = 0 \quad &\text{on } \Gamma \quad \text{(Homogeneous Dirichlet)}, \\ \Sigma_{lin} \cdot n = 0 \quad &\text{on } \partial\Omega \setminus \Gamma \quad \text{(Homogeneous Neumann)}. \end{aligned}$$

11.4.1.3 Material parameters and given data The material parameters are:

```
lame_coefficient_mu = 1.0e+6;
poisson_ratio_nu = 0.4;

lame_coefficient_lambda =  (2 * poisson_ratio_nu * lame_coefficient_mu)/
  (1.0 - 2 * poisson_ratio_nu);
```

and the force vector is prescribed as

$$f(x, y, z) = (0, -9.81, 0)^T.$$

11.4.1.4 Simulation results With a PCG scheme (see Section 10.8), we need 788 PCG iterations to obtain convergence of the linear solver. The resulting solution is displayed in Figure 42.

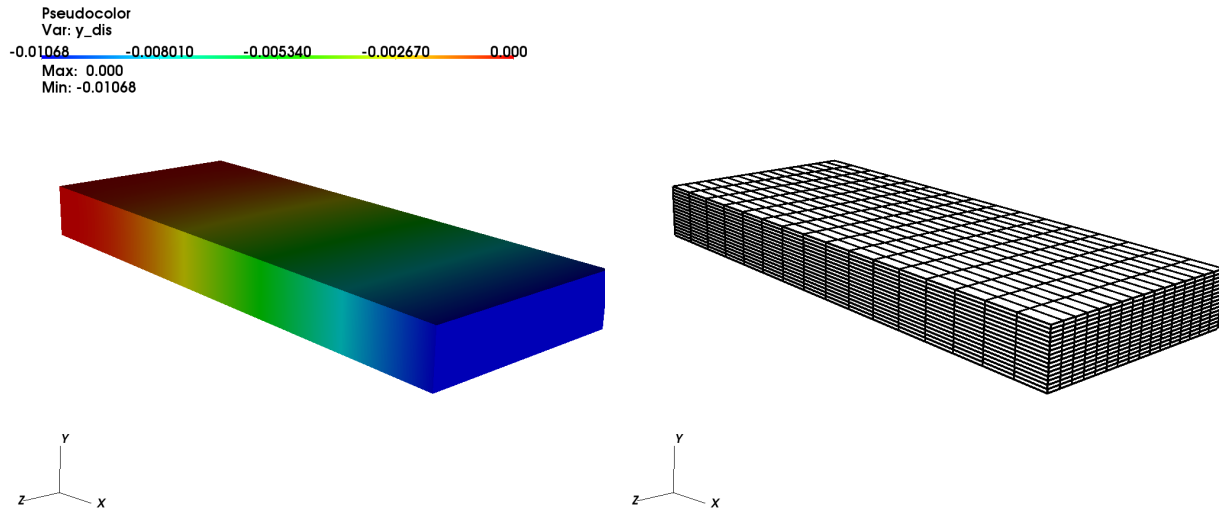


Figure 42: 3D linearized elasticity: u_y solution and mesh. The elastic beam is attached at $x = 0$ in the $y - z$ plane by $u_x = u_y = u_z = 0$. All other boundaries are traction free (homogeneous Neumann conditions).

11.4.2 2D test with focus on the maximum principle

In the corresonding 2D test (because it is easier to show), we demonstrate the violation of the maximum principle.

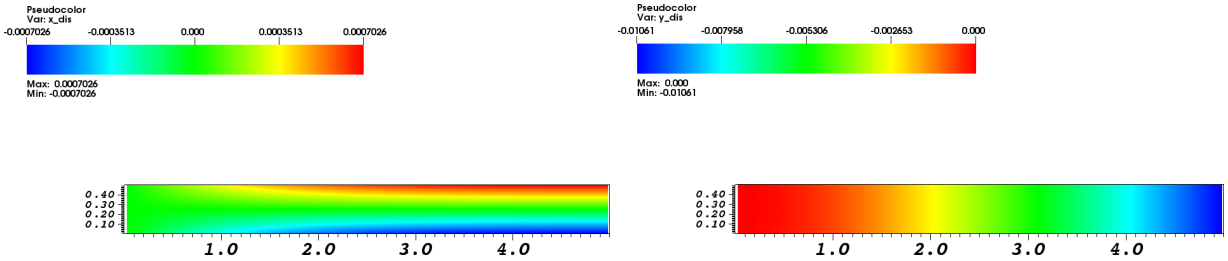


Figure 43: 2D linearized elasticity. The same test as for 3D. At left, the u_x solution is displayed. At right, the u_y solution is displayed.

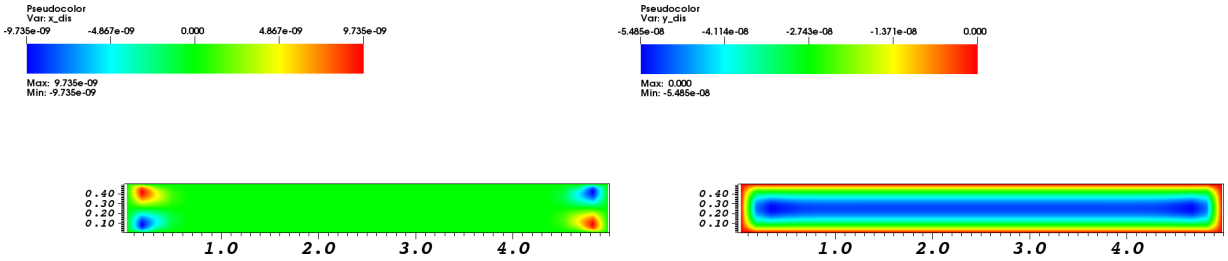


Figure 44: 2D linearized elasticity: all boundaries are subject to homogeneous Dirichlet conditions. In the u_x solution (left), we observe that the minimum and maximum are obtained inside the domain, which is a violation of the maximum principle.

11.5 Stokes - FEM discretization

In Section 9.24, we have formulated the stationary NSE system. Without convection term, we arrive at the Stokes system for viscous flow (such as honey for instance). Without going into any theoretical details, we briefly present the finite element scheme in the following. A classical reference to NSE FEM discretizations is [57].

Let $U_h = \{v_h, p_h\}$, $X_h := V_h \times L_h = H_0^1 \times L_0$ and $\Psi = \{\psi^v, \psi^p\}$. The problem reads:

$$\text{Find } U_h \in X_h \text{ such that: } A(U_h, \Psi_h) = F(\Psi_h) \quad \forall \Phi \in X_h,$$

where

$$\begin{aligned} A(U_h, \Psi_h) &= (\nabla v_h, \nabla \psi_h^v) - (p_h, \nabla \cdot \psi_h^v) + (\nabla \cdot v_h, \psi_h^p), \\ F(\Psi_h) &= (f_f, \psi_h^v). \end{aligned}$$

Let us choose as basis:

$$\begin{aligned} V_h &= \{\psi_h^{v,i}, i = 1, \ldots, N_V := dim V_h\}, \\ L_h &= \{\psi_h^{p,i}, i = 1, \ldots, N_P := dim L_h\}. \end{aligned}$$

Please pay attention that $V_h := (V_h)^d$ is a vector-valued space with dimension d. If follows:

$$\begin{aligned} (\nabla v_h, \nabla \psi_h^{v,i}) - (p_h, \nabla \cdot \psi_h^{v,i}) \quad & i = 1, \ldots, N_V, \\ (\nabla \cdot v_h, \psi_h^{p,i}) \quad & i = 1, \ldots, N_P, \end{aligned}$$

Setting:

$$v_h = \sum_{j=1}^{N_V} v_j \psi_h^{v,j}, \quad p_h = \sum_{j=1}^{N_P} p_j \psi_h^{p,j}$$

yield the discrete equations:

$$\begin{aligned} \sum_{j=1}^{N_V} (\nabla \psi_h^{v,j}, \nabla \psi_h^{v,i}) v_j - \sum_{j=1}^{N_P} (\psi_h^{p,j}, \nabla \cdot \psi_h^{v,i}) p_j, \quad & i = 1, \ldots, N_V, \\ \sum_{j=1}^{N_v} (\nabla \cdot \psi_h^{v,j}, \psi_h^{p,i}), \quad & i = 1, \ldots, N_P. \end{aligned}$$

With this, we obtain the following matrices:

$$A := (\nabla \psi_h^{v,j}, \nabla \psi_h^{v,i})_{ij=1}^{N_V, N_V}, \quad B := -(\psi_h^{p,j}, \nabla \cdot \psi_h^{v,i})_{ij=1}^{N_V, N_P}, \quad -B^T = (\nabla \cdot \psi_h^{v,j}, \psi_h^{p,i})_{ij=1}^{N_P, N_V}.$$

These matrices form the block system:

$$\begin{pmatrix} A & B \\ -B^T & 0 \end{pmatrix} \begin{pmatrix} v \\ p \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix}, \tag{233}$$

where $v = (v_i)_{i=1}^{N_V}, p = (p_i)_{i=1}^{N_P}$.

11.6 Numerical test - 2D-1 benchmark without convection term (Stokes)

We take the same material parameters as in Section 9.24, but now without the convection term. The results are displayed in Figure 45.

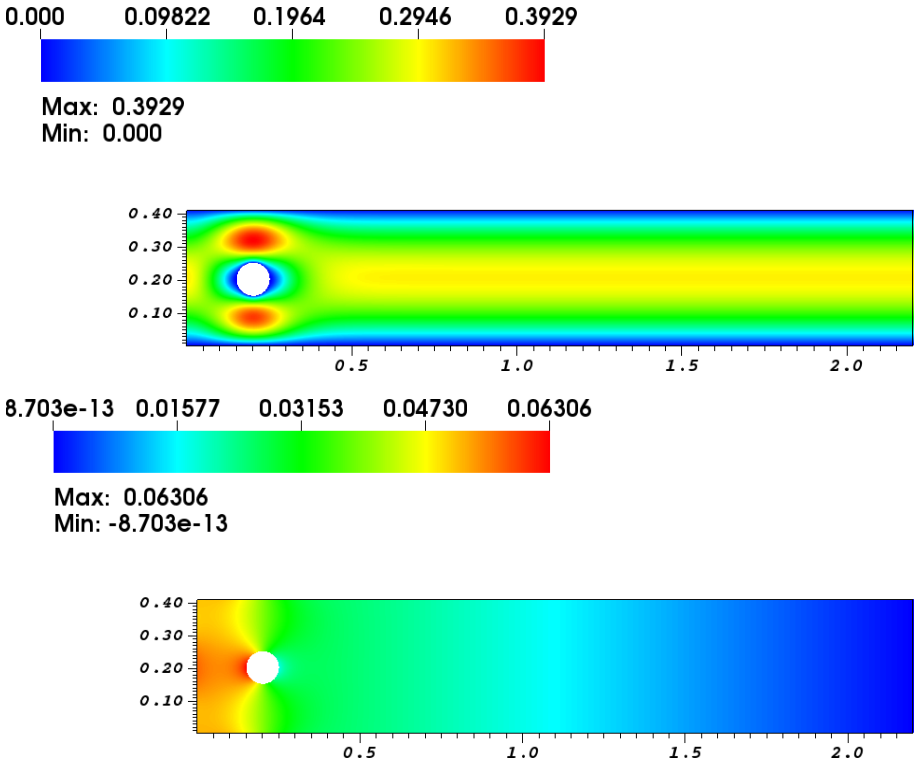


Figure 45: Stokes 2D-1 test: x velocity profile and pressure distribution.

The functional values on a three times uniformly-refined mesh are:

```
==================================
P-Diff:      4.5559514861737337e-02
P-front:     6.3063021215059634e-02
P-back:      1.7503506353322297e-02
Drag:        3.1419886302140707e+00
Lift:        3.0188981539007183e-02
==================================
```

11.7 Chapter summary and outlook

In this chapter, we extended from scalar-valued elliptic problems to vector-valued elliptic problems. The well-posedness analysis follows the same lines as the scalar cases. We notice that the maximum principle does not hold anymore in the vector-valued case (not proven though, but shown in a numerical example.). The finite element formulation is in principle the same as the scalar case, but we have to work with vector-valued function spaces now. The formal procedure is the same as before, but the computational cost increases significantly since now three variables need to discretized (in 3D) rather than only one variable.

12 Methods for time-dependent PDEs: parabolic and hyperbolic problems

By now, we have considered **stationary, linear** PDE problems. However, more realistic, and most notably fascinating, modeling is

- nonstationary,
- and nonlinear.

We investigate the former one in the present chapter and give a brief introduction to nonlinear problems in Chapter 13.

Therefore, the focus of this chapter is on

- time-dependent linear PDEs.

We concentrate first on parabolic PDEs and later study second-order-in-time hyperbolic PDEs. For first-order hyperbolic PDEs, we refer to [82].

A prominent example of a parabolic PDE is the heat equation that we investigate in this section. In the next section, we consider the second-order hyperbolic wave equation.

12.1 Principle procedures for discretizing time and space

Time-dependent PDE problems require require discretization in space and time. One can mainly distinguish three procedures:

- Vertical method of lines (method of lines): first space, then time.
- Horizontal method of lines (Rothe method): first time, then space.
- A full space-time Galerkin method.

Using one of the first two methods, and this is also the procedure we shall follow here, temporal discretization is based on FD whereas spatial discretization is based on the FEM. In the following we concentrate on the Rothe method and we briefly provide reasons why we want to work with the Rothe method. In fact, the traditional way of discretizing time-dependent problems is the (vertical) method of lines. The advantage of the method of lines is that we have simple data structures and matrix assembly. This leads to a (large) ODE system, which can be treated by (well-known) standard methods from ODE analysis (ordinary differential equations) can be employed for time discretization. The major disadvantage is that the spatial mesh is fixed (since we first discretize in space) and it is then difficult to represent or compute time-varying features such as certain target functionals (e.g., drag or lift).

In contrast, the Rothe method allows for dynamic spatial mesh adaptation with the price that data structures and matrix assembly are more costly.

12.2 Bochner spaces - space-time functions

For the correct function spaces for formulating time-dependent variational forms, we define the Bochner integral. Let $I := (0,T)$ with $0 < T < \infty$ a bounded time interval with end time value T. For any Banach space X and $1 \leq p \leq \infty$, the space

$$L^p(I,X)$$

denotes the space of L^p integrable functions f from the time interval I into X. This is a Banach space, the so-called Bochner space, with the norm, see [145],

$$\|v\|_{L^p(I,X)} := \Big(\int_I \|v(t)\|_X^p \, dt\Big)^{1/p}$$
$$\|v\|_{L^\infty(I,X)} := \operatorname*{ess\,sup}_{t\in I} \|v(t)\|_X.$$

For the definition of the norms of the spatial spaces, i.e., $\|v(t)\|_X$ with $X = L^2$ or $X = H^1$, we refer to Section 7.1.7.

Example 12.1. *For instance, we can define a H^1 space in time:*

$$H^1(I,X) = \left\{ v \in L^2(I,X) |\, \partial_t v \in L^2(I,X) \right\}.$$

Functions that are even continuous in time, i.e., $u : I \to X$, are contained in spaces like

$$C(I;X)$$

with

$$\|u\|_{C(I;X)} := \max_{0\le t\le T} \|u(t)\| < \infty.$$

Definition 12.2 (Weak derivative of space-time functions)**.** *Let $u \in L^1(I;X)$. A function $v \in L^1(I;X)$ is the weak derivative of v, denoted as*

$$\partial_t u = v$$

if

$$\int_0^T \partial_t \varphi(t) u(t)\, dt = -\int_0^T \varphi(t) v(t)\, dt$$

for all test functions $\varphi \in C_c^\infty(I)$.

In particular, the following result holds:

Theorem 12.3 ([50])**.** *Assume $v \in L^2(I, H_0^1)$ and $\partial_t v \in L^2(I, H^{-1})$. Then, v is continuous in time, i.e.,*

$$v \in C(I, L^2)$$

(after possible redefined on a set of measure zero). Furthermore, the mapping

$$t \mapsto \|v(t)\|^2_{L^2(X)}$$

is absolutely continuous with

$$\frac{d}{dt}\|v(t)\|^2_{L^2(X)} = 2\langle \frac{d}{dt} v(t), v(t)\rangle$$

for a.e. $0 \le t \le T$.

Proof. See Evans [50], Theorem 3 in Section 5.9.2. □

The importance of this theorem lies in the fact that now the point-wise prescription of initial conditions does make sense in weak formulations.

Remark 12.4. *More details of these spaces by means of the Bochner integral can be found in [39, 144] and also [50].*

12.3 Methods for parabolic problems

To illustrate the concepts of discretizing we work in 1D (one spatial dimension) in the following.

12.3.1 Problem statements

Let us consider the following abstract problem:

Formulation 12.5. *Consider the following parabolic problem: Find u such that:*

$$\begin{aligned} \partial_t u + A(u) &= f \quad \text{in } \Omega \times I, \\ u(x,t) &= u_D \quad \text{on } \partial\Omega \times I, \\ u(x,0) &= u^0 \quad \text{in } \partial\Omega \times \{0\}, \end{aligned}$$

where A is a linear elliptic operator.

A concrete realization is the following situation: Let Ω be an open, bounded subset of $\mathbb{R}^d, d = 1$ and $I := (0, T]$ where $T > 0$ is the end time value. The IBVP (initial boundary-value problem) reads:

Formulation 12.6. *Find $u := u(x,t) : \Omega \times I \to \mathbb{R}$ such that*

$$\begin{aligned} \rho\partial_t u - \nabla \cdot (\alpha \nabla u) &= f \quad \text{in } \Omega \times I, \\ u &= a \quad \text{on } \partial\Omega \times [0,T], \\ u(0) &= g \quad \text{in } \Omega \times t = 0, \end{aligned}$$

where $f : \Omega \times I \to \mathbb{R}$ and $g : \Omega \to \mathbb{R}$ and $\alpha \in \mathbb{R}$ and ρ are material parameters, and $a \geq 0$ is a Dirichlet boundary condition. More precisely, g is the initial temperature and a is the wall temperature, and f is some heat source.

For the variational formulation of the abstract form, let H and V be Hilbert spaces with V^* being the dual space of V. A Gelfand triple is denoted by

$$V \hookrightarrow H \hookrightarrow V^*$$

Definition 12.7 (Inner products)**.** *The spatial inner product on H is denoted by*

$$(u, \varphi)_H = \int_\Omega u \cdot \varphi \, dx$$

The temporal inner product on a space-time space X is denoted by

$$(u, \varphi) := (u, \varphi)_X = \int_0^T (u, \varphi)_H \, dt$$

Let us now define the function spaces. In space, we choose our well-known canditates:

$$V := H_0^1(\Omega), \quad H := L^2(\Omega)$$

On the time interval I, we introduce the Hilbert space $X := W(0,T)$ with

$$W(0,T) := \{v | v \in L^2(I,V),\ \partial_t v \in L^2(I,V^*)\}$$

The notation $L^2(I,V)$ means that v is square-integrable in time using values from I and mapping to the image space V. The space X is embedded in $C(\bar{I}, H)$, which allows to work with well-defined (continuous) initial conditions u^0.

Definition 12.8 (Bilinear forms)**.** *We define the spatial bilinear form $\bar{a} : V \times V \to \mathbb{R}$ as usually. The time-dependent semi-linear form is as follows: $a : X \times X \to \mathbb{R}$:*

$$a(u, \varphi) := \int_0^T \bar{a}(u(t), \varphi(t)) \, dt.$$

Formulation 12.9 (Space-time weak form)**.** *Then the space-time parabolic problem is given by: Find $u \in X$ such that*

$$\begin{aligned} (\partial_t u, \varphi) + a(u, \varphi) &= (f, \varphi) \quad \forall \varphi \in X \\ u(0) &= u^0 \end{aligned}$$

with $f \in L^2(I, V^*)$ and $u^0 \in H$.

12.3.2 Temporal discretization via One-Step-θ schemes

We first discretize in time and create a time grid of the time domain $I = [0, T]$ with N_T intervals and a time step size $k = \frac{T}{N_T}$:

$$0 = t_0 < t_1 < \ldots < t_N = T.$$

Furthermore we set

$$I_n = [t_{n-1}, t_n], \quad k_n = t_n - t_{n-1}, \quad k := \max_{1 \le n \le N} k_n.$$

Moreover, we denote $u^n := u(t^n)$.

The simplest discretizations are explicit and implicit Euler:

Formulation 12.10 (Explicit and implicit Euler)**.** *Given u^{n-1}, find u^n such that*

$$\frac{u^n - u^{n-1}}{k} - \nabla \cdot (\alpha \nabla u^{n-1}) = f^{n-1}$$

and for the implicit Euler scheme:

$$\frac{u^n - u^{n-1}}{k} - \nabla \cdot (\alpha \nabla u^{n}) = f^{n}$$

Here we use finite differences and more specifically we introduce the so-called **One-Step-θ** scheme, which allows for a compact notation for three major finite difference schemes: forward Euler ($\theta = 0$), backward Euler ($\theta = 1$), and Crank-Nicolson ($\theta = 0.5$) well known from numerical methods for ODEs.

Definition 12.11 (Choice of θ)**.** *By the choice of θ, we obtain the following time-stepping schemes:*

- $\theta = 0$*: 1st order explicit Euler time stepping;*
- $\theta = 0.5$*: 2nd order Crank-Nicolson (trapezoidal rule) time stepping;*
- $\theta = 0.5 + k_n$*: 2nd order shifted Crank-Nicolson which is shifted by the time step size $k_n = t^n - t^{n-1}$ towards the implicit side;*
- $\theta = 1$*: 1st order implicit Euler time stepping.*

To have good stability properties (namely A-stability) of the time-stepping scheme is important for temporal discretization of partial differential equations. Often (as here for the heat equation) we deal with second-order operators in space, such PDE-problems are generically (very) stiff with the order $O(h^{-2})$, where h is the spatial discretization parameter.

After these preliminary considerations, let us now discretize in time the above problem:

Formulation 12.12 (One-Step-θ for the heat equation)**.** *The one-step-θ scheme for the heat equation reads: Given the continuous PDE problem:*

$$\partial_t u - \nabla \cdot (\alpha \nabla u) = f,$$

we obtain

$$\frac{u^n - u^{n-1}}{k} - \theta \nabla \cdot (\alpha \nabla u^n) - (1-\theta) \nabla \cdot (\alpha \nabla u^{n-1}) = \theta f^n + (1-\theta) f^{n-1}$$

Re-organizing in known and unknown terms, we obtain:

$$\underbrace{u^n - k\theta \nabla \cdot (\alpha \nabla u^n)}_{unknown} = \underbrace{u^{n-1} + k(1-\theta) \nabla \cdot (\alpha \nabla u^{n-1}) + k\theta f^n + k(1-\theta) f^{n-1}}_{known}$$

where $u^n := u(t^n)$ and $u^{n-1} := u(t^{n-1})$ and $f^n := f(t^n)$ and $f^{n-1} := f(t^{n-1})$.

12.3.3 Spatial discretization and an abstract One-Step-θ scheme

We take the temporally discretized problem and use a Galerkin finite element scheme to discretize in space as explained in Section 8. That is we multiply with a test function and integrate. We then obtain: Find $u_h \in \{a + V_h\}$ such that for $n = 1, 2, \dots N_T$:

$$(u_h^n, \varphi_h) - k\theta(\alpha \nabla u_h^n, \nabla \varphi_h) = (u^{n-1}, \varphi_h) + k(1-\theta)(\alpha \nabla u^{n-1}, \nabla \varphi_h) + k\theta(f^n, \varphi_h) + k(1-\theta)(f^{n-1}, \varphi_h)$$

for all $\varphi_h \in V_h$.

Definition 12.13 (Specification of the problem data)**.** *For simplicity let us assume that there is no heat source $f = 0$ and the material coefficients are specified by $\alpha = 1$ and $\rho = 1$.*

With the help of the semi-linear forms, we can formulate an abstract time-stepping scheme.

Formulation 12.14 (Abstract one-step-θ scheme)**.** *The one-step-θ scheme can be formulated in the following abstract way: Given $u^{n-1} \in V$, find $u^n \in V$ such that*

$$a_T(u^{n,k}, \varphi) + \theta a_E(u^n, \varphi) = -(1-\theta) a_E(u^{n-1}, \varphi) + \theta l^n(\varphi) + (1-\theta) l^{n-1}(\varphi).$$

Here:

$$\begin{aligned} a_T(u^{n,k}, \varphi) &= \frac{1}{k}(u^n - u^{n-1}, \varphi) \\ a_E(u^n, \varphi) &= (\alpha \nabla u^n, \nabla \varphi) \\ l^n(\varphi) &= (f^n, \varphi). \end{aligned}$$

12.3.4 Spatial discretization for the forward Euler scheme

Let us work with the explicit Euler scheme $\theta = 0$. Then we obtain:

$$(u_h^n, \varphi_h) = (u_h^{n-1}, \varphi_h) - k(\nabla u_h^{n-1}, \nabla \varphi_h)$$

The structure of this equation, namely

$$y^n = y^{n-1} + k f(t^{n-1}, y^{n-1}),$$

is the same as we know from ODEs. On the other hand, the single terms in the weak form are akin to our derivations in Section 8. This means that we seek at each time t^n a finite element solution u_h^n on the spatial domain Ω. With the help of the basis functions $\varphi_{h,j}$ we can represent the spatial solution:

$$u_h^n(x) := \sum_{j=1}^{N_x} u_{h,j} \varphi_{h,j}, \quad u_{h,j} \in \mathbb{R}.$$

Then:

$$\sum_{j=1}^{N_x} u_{h,j}^n (\varphi_{h,j}, \varphi_{h,i}) = (u^{n-1}, \varphi_{h,i}) - k(\nabla u_h^{n-1}, \nabla \varphi_{h,i})$$

which results in a linear equation system: $MU = B$. Here

$$\begin{aligned} (M_{ij})_{i,j=1}^{N_x} &:= (\varphi_{h,j}, \varphi_{h,i}), \\ (U_j)_{j=1}^{N_x} &:= (u_{h,1}, \dots, u_{h,N}) \\ (B_i)_{i=1}^{N_x} &:= (u_h^{n-1}, \varphi_{h,i}) - k(\nabla u_h^{n-1}, \nabla \varphi_{h,i}). \end{aligned}$$

Furthermore the old time step solution can be written as well in matrix form:

$$(u^{n-1}, \varphi_{h,i}) = \sum_{j=1}^{N_x} u_{h,j}^{n-1} \underbrace{(\varphi_{h,j}, \varphi_{h,i})}_{=:M}$$

and similarly for the Laplacian term:

$$(\nabla u^{n-1}, \nabla \varphi_{h,i}) = \sum_{j=1}^{N_x} u_{h,j}^{n-1} \underbrace{(\nabla \varphi_{h,j}, \nabla \varphi_{h,i})}_{=:K}.$$

The mass matrix M (also known as the Gramian matrix) is always the same for fixed h and can be explicitely computed.

Proposition 12.15. *Formally we arrive at: Given u_h^{n-1}, find u_h^n, such that*

$$Mu_h^n = Mu_h^{n-1} - kKu_h^{n-1} \quad \Rightarrow \quad u_h^n = u_h^{n-1} - kM^{-1}Ku_h^{n-1}.$$

for $n = 1, \dots, N_T$.

Remark 12.16. *We emphasize that the forward Euler scheme is not recommended to be used as time stepping scheme for solving PDEs. The reason is that the stiffness matrix is of order $\frac{1}{h^2}$ and one is in general interested in $h \to 0$. Thus the coefficients become very large for $h \to 0$ resulting in a stiff system. For stiff systems, as we learned before, one should better use implicit schemes, otherwise the time step size k has to be chosen too small in order to obtain stable numerical results; see the numerical analysis in Section 12.3.10.*

12.3.5 Evaluation of the integrals (1D in space)

We need to evaluate the integrals for the stiffness matrix K:

$$K = (K_{ij})_{ij=1}^{N_x} = \int_\Omega \varphi'_{h,j}(x)\varphi'_{h,i}(x)\,dx = \int_{x_{j-1}}^{x_{j+1}} \varphi'_{h,j}(x)\varphi'_{h,i}(x)\,dx$$

resulting in

$$A = h^{-1} \begin{pmatrix} 2 & -1 & & & 0 \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ 0 & & & -1 & 2 \end{pmatrix}$$

Be careful and do not forget the material parameter α in case it is not $\alpha = 1$.

For the mass matrix M we obtain:

$$M = (M_{ij})_{ij=1}^{N_x} = \int_\Omega \varphi_{h,j}(x)\varphi_{h,i}(x)\,dx = \int_{x_{j-1}}^{x_{j+1}} \varphi_{h,j}(x)\varphi_{h,i}(x)\,dx.$$

Specifically on the diagonal, we calculate:

$$\begin{aligned} M_{ii} &= \int_\Omega \varphi_{h,i}(x)\varphi_{h,i}(x)\,dx = \int_{x_{i-1}}^{x_i} \Big(\frac{x - x_{i-1}}{h}\Big)^2 dx + \int_{x_i}^{x_{i+1}} \Big(\frac{x_{i+1} - x}{h}\Big)^2 dx \\ &= \frac{1}{h^2}\int_{x_{i-1}}^{x_i} (x - x_{i-1})^2\,dx + \frac{1}{h^2}\int_{x_i}^{x_{i+1}} (x_{i+1} - x)^2\,dx = \frac{h}{3} + \frac{h}{3} \\ &= \frac{2h}{3}. \end{aligned}$$

For the right off-diagonal, we have

$$\begin{aligned} m_{i,i+1} &= \int_\Omega \varphi_{h,i+1}(x)\varphi_{h,i}(x)\,dx = \int_{x_i}^{x_{i+1}} \frac{x - x_i}{h} \cdot \frac{x_{i+1} - x}{h}\,dx = \int_{x_i}^{x_{i+1}} \frac{x - x_i}{h} \cdot \frac{x_i - x + h}{h}\,dx \\ &= \dots \\ &= -\frac{h}{3} + \frac{h^2}{2h} = \frac{h}{6}. \end{aligned}$$

It is trivial to see that $m_{i,i+1} = m_{i-1,i}$. Summarizing all entries results in

$$M = \frac{h}{6}\begin{pmatrix} 4 & 1 & & & 0 \\ 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 4 & 1 \\ 0 & & & 1 & 4 \end{pmatrix}.$$

12.3.6 Final algorithms

We first setup a sequence of discrete (time) points:

$$0 = t_0 < t_1 < \ldots < t_N = T.$$

Furthermore we set

$$I_n = [t_{n-1}, t_n], \quad k_n = t_n - t_{n-1}, \quad k := \max_{1 \le n \le N} k_n.$$

Forward (explicit) Euler:

Algorithm 12.17. *Given the initial condition g, solve for $n = 1, 2, 3, \ldots, N_T$*

$$Mu_h^n = Mu_h^{n-1} - kKu_h^{n-1} \quad \Rightarrow \quad u_h^n = u_h^{n-1} - kM^{-1}Ku_h^{n-1},$$

where $u_h^n, u_h^{n-1} \in \mathbb{R}^{N_x}$.

Algorithm 12.18. *The backward (implicit) Euler scheme reads:*

$$Mu_h^n + kKu_h^n = Mu_h^{n-1}$$

An example of a pseudo C++ code is

```
final_loop()
{
 // Initalization of model parameters, material parameters, and so forth
 set_runtime_parameters();

 // Make a mesh - decompose the domain into elements
 create_mesh();

 apply_initial_conditions();

 // Timestep loop
  do
    {
      std::cout << "Timestep " << timestep_number << " (" << time_stepping_scheme
                << ")" <<    ": " << time << " (" << timestep << ")"
                << "\n=============================================================================="
                << std::endl;

      std::cout << std::endl;

      // Solve for next time step: Assign previous time step solution u^{n-1}
      old_timestep_solution = solution;

      // Assemble FEM matrix A and right hand side f
      assemble_system ();
```

```
      // Solve the linear equation system Ax=b
      solve ();

      // Update time
      time += timestep;

      // Write solution into a file or similar
      output_results (timestep_number);

      // Increment n->n+1
      ++timestep_number;
    }
  while (timestep_number <= max_no_timesteps);
}
```

12.3.7 Recapitulation of ODE stability analysis using finite differences

This section is optional. We recall the stability analysis for ODEs. The principle ideas are exactly the same as in Section 6.

12.3.7.1 The Euler method The Euler method is the most simplest scheme. It is an explicit scheme and also known as forward Euler method.

We consider again the IVP from before and the right hand side satisfies again a Lipschitz condition. According to ODE theory (e.g., [103]), there exists a unique solution for all $t \geq 0$.

For a numerical approximation of the IVP, we first select a sequence of discrete (time) points:

$$t_0 < t_1 < \ldots < t_N = t_0 + T.$$

Furthermore we set

$$I_n = [t_{n-1}, t_n], \quad k_n = t_n - t_{n-1}, \quad k := \max_{1 \leq n \leq N} k_n.$$

The derivation of the Euler method is as follows: approximate the derivative with a forward difference quotient (we sit at the time point t_{n-1} and look forward in time):

$$y'(t_{n-1}) \approx \frac{y_n - y_{n-1}}{k_n}$$

Thus: $y'(t_{n-1}) = f(t_{n-1}, y_{n-1}(t_{n-1}))$. Then the ODE can be approximated as:

$$\frac{y_n - y_{n-1}}{k_n} \approx f(t_{n-1}, y_{n-1}(t_{n-1}))$$

Thus we obtain the scheme:

Algorithm 12.19 (Euler method)**.** *For a given starting point $y_0 := y(0) \in \mathbb{R}^n$, the Euler method generates a sequence $\{y_n\}_{n \in \mathbb{N}}$ through*

$$y_n = y_{n-1} + k_n f(t_{n-1}, y_{n-1}), \quad n = 1, \ldots N,$$

where $y_n := y(t_n)$.

Remark 12.20 (Notation)**.** *The chosen notation is not optimal in the sense that y_n denotes the discrete solution obtained by the numerical scheme and $y(t_n)$ the corresponding (unknown) exact solution. However, in the literature one often abbreviates $y_n := y(t_n)$, which would both denote the exact solution. One could add another index y_n^k (k for discretized solution with step size k) to explicitely distinguish the discrete and exact solutions. In these notes, we hope that the reader will not confuse the notation and we still use y_n for the discrete solution and $y(t_n)$ for the exact solution.*

12.3.7.2 Implicit schemes With the same notation as introduced in Section 12.3.7.1, we define two further schemes. Beside the Euler method, low-order simple schemes are implicit Euler and the trapezoidal rule. The main difference is that in general a nonlinear equation system needs to be solved in order to compute the solution. On the other hand we have better numerical stability properties in particular for stiff problems (an analysis will be undertaken in Section 12.3.7.3).

The derivation of the backward Euler method is derived as follows: approximate the derivative with a backward difference quotient (we sit at t_n and look back to t_{n-1}):

$$y'(t_n) = \frac{y_n - y_{n-1}}{k_n}$$

Consequently, we take the right hand side f at the current time step $y'(t_n) = f(t_n, y_n(t_n))$ and obtain as approximation

$$\frac{y_n - y_{n-1}}{k_n} = f(t_n, y_n(t_n)).$$

Consequently, we obtain a scheme in which the right hand side is unknown itself, which leads to a formulation of a nonlinear system:

Algorithm 12.21 (Implicit (or backward) Euler)**.** *The implicit Euler scheme is defined as:*

$$\begin{aligned} y_0 &:= y(0), \\ y_n - k_n f(t_n, y_n) &= y_{n-1}, \quad n = 1, \dots, N \end{aligned}$$

In contrast to the Euler method, the 'right hand side' function f now depends on the unknown solution y_n. Thus the computational cost is (much) higher than for the (forward) Euler method. But on the contrary, the method does not require a time step restriction as we shall see in Section 12.3.7.4.

To derive the trapezoidal rule, we take again the difference quotient on the left hand side but approximate the right hand side through its mean value:

$$\frac{y_n - y_{n-1}}{k_n} = \frac{1}{2}\Big(f(t_n, y_n(t_n)) + f(t_{n-1}, y_{n-1}(t_{n-1}))\Big),$$

which yields:

Algorithm 12.22 (Trapezoidal rule (Crank-Nicolson))**.** *The trapezoidal rule reads:*

$$\begin{aligned} y_0 &:= y(0), \\ y_n &= y_{n-1} + \frac{1}{2} k_n \Big(f(t_n, y_n) + f(t_{n-1}, y_{n-1}\Big), \quad n = 1, \dots, N. \end{aligned}$$

It can be shown that the trapezoidal rule is of second order, which means that halving the step size k_n leads to an error that is four times smaller.

12.3.7.3 Numerical analysis In the previous section, we have constructed algorithms that yield a sequence of discrete solution $\{y_n\}_{n\in\mathbb{N}}$. In the numerical analysis our goal is to derive a convergence result of the form

$$\|y_n - y(t_n)\| \leq Ck^\alpha$$

where α is the order of the scheme. This result will tell us that the discrete solution y^n really approximates the exact solution y and if we come closer to the exact solution at which rate we come closer. To derive error estimates we work with the model problem:

$$y' = \lambda y, \quad y(0) = y_0, \quad y_0, \lambda \in \mathbb{R}. \tag{234}$$

For linear numerical schemes the convergence is composed by **stability** and **consistency**.

First of all we have from the previous section that y_n is obtained for the forward Euler method as:

$$\begin{aligned} y_n &= (1 + k\lambda) y_{n-1}, && (235) \\ &= B_E y_{n-1}, \quad B_E := (1 + k\lambda). && (236) \end{aligned}$$

Let us write the error at each time point t_n as:

$$e_n := y_n - y(t_n) \quad \text{for } 1 \leq n \leq N.$$

It holds:

$$\begin{aligned} e_n &= y_n - y(t_n), \\ &= B_E y_{n-1} - y(t_n), \\ &= B_E(e_{n-1} + y(t_{n-1})) - y(t_n), \\ &= B_E e_{n-1} + B_E y(t_{n-1}) - y(t_n), \\ &= B_E e_{n-1} + \frac{k(B_E y(t_{n-1}) - y(t_n))}{k}, \\ &= B_E e_{n-1} - k \underbrace{\frac{y(t_n) - B_E y(t_{n-1})}{k}}_{=:\eta_{n-1}}. \end{aligned}$$

Therefore, the error can be split into two parts:

Definition 12.23 (Error splitting of the model problem)**.** *The error at step n can be decomposed as*

$$e_n := \underbrace{B_E e_{n-1}}_{Stability} - \underbrace{k\eta_{n-1}}_{Consistency}. \tag{237}$$

The first term, namely the stability, provides an idea how the previous error e_{n-1} is propagated from t_{n-1} to t_n. The second term η_{n-1} is the so-called truncation error (or local discretization error), which arises because the exact solution does not satisfy the numerical scheme and represents the consistency of the numerical scheme. Moreover, η_{n-1} yields the speed of convergence of the numerical scheme.

In fact, for the forward Euler scheme in (237), we observe for the truncation error:

$$\eta_{n-1} = \frac{y(t_n) - B_E y(t_{n-1})}{k} = \frac{y(t_n) - (1 + k\lambda)y(t_{n-1})}{k} \tag{238}$$

$$= \frac{y(t_n) - y(t_{n-1})}{k} - \lambda y(t_{n-1}) \tag{239}$$

$$= \frac{y(t_n) - y(t_{n-1})}{k} - y'(t_{n-1}). \tag{240}$$

Thus,

$$y'(t_{n-1}) = \frac{y(t_n) - y(t_{n-1})}{k} - \eta_{n-1}, \tag{241}$$

which is nothing else than the approximation of the first-order derivative with the help of a difference quotient plus the truncation error. We investigate these terms further in Section 12.3.7.5 and concentrate first on the stability estimates in the very next section.

12.3.7.4 Stability The goal of this section is to control the term $B_E = (1 + k\lambda)$. Specifically, we will justify why $|B_E| \leq 1$ should hold. The stability is often related to non-physical oscillations of the numerical solution.

We recapitulate (absolute) **stability** and **A-stability**. From the model problem

$$y'(t) = \lambda y(t), \quad y(t_0) = y_0, \ \lambda \in \mathbb{C},$$

we know the solution $y(t) = y_0 \exp(\lambda t)$. For $t \to \infty$ the solution is characterized by the sign of $Re\,\lambda$:

$$\begin{aligned} Re\,\lambda < 0 \quad &\Rightarrow |y(t)| = |y_0| \exp(Re\,\lambda) \to 0, \\ Re\,\lambda = 0 \quad &\Rightarrow |y(t)| = |y_0| \exp(Re\,\lambda) = |y_0|, \\ Re\,\lambda > 0 \quad &\Rightarrow |y(t)| = |y_0| \exp(Re\,\lambda) \to \infty. \end{aligned}$$

For a *good* numerical scheme, the first case is particularly interesting whether such a scheme can produce a bounded discrete solution when the continuous solution has this property.

Definition 12.24 ((Absolute) stability)**.** *A (one-step) method is absolute stable for $\lambda k \neq 0$ if its application to the model problem produces in the case $Re\,\lambda \leq 0$ a sequence of bounded discrete solutions: $\sup_{n\geq 0}|y_n| < \infty$. To find the stability region, we work with the stability function $R(z)$ where $z = \lambda k$. The region of absolute stability is defined as:*

$$SR = \{z = \lambda k \in \mathbb{C} : |R(z)| \leq 1\}.$$

Remark 12.25. *Recall that $R(z) := B_E$.*

The stability functions to explicit, implicit Euler and trapezoidal rule are given by:

Proposition 12.26. *For the simplest time-stepping schemes forward Euler, backward Euler and the trapezoidal rule, the stability functions $R(z)$ read:*

$$\begin{aligned} R(z) &= 1 + z, \\ R(z) &= \frac{1}{1-z}, \\ R(z) &= \frac{1+\frac{1}{2}z}{1-\frac{1}{2}z}. \end{aligned}$$

Proof. We take again the model problem $y' = \lambda y$. Let us discretize this problem with the forward Euler method:

$$\begin{aligned} \frac{y_n - y_{n-1}}{k} &= \lambda y_{n-1} && (242)\\ \Rightarrow y_n &= (y_{n-1} + \lambda k) y_{n-1} && (243)\\ &= (1 + \lambda k) y_{n-1} && (244)\\ &= (1 + z) y_{n-1} && (245)\\ &= R(z) y_{n-1}. && (246) \end{aligned}$$

For the implicit Euler method we obtain:

$$\begin{aligned} \frac{y_n - y_{n-1}}{k} &= \lambda y_n && (247)\\ \Rightarrow y_n &= (y_{n-1} + \lambda k) y_n && (248)\\ \Rightarrow y_n &= \frac{1}{1-ak} y_n && (249)\\ \Rightarrow y_n &= \underbrace{\frac{1}{1-z}}_{=:R(z)} y_n. && (250) \end{aligned}$$

The procedure for the trapezoidal rule is again the analogous. □

Definition 12.27 (A-stability)**.** *A difference method is A-stable if its stability region is part of the absolute stability region:*

$$\{z \in \mathbb{C} : Re\, z \leq 0\} \subset SR,$$

here Re denotes the real part of the complex number z. A brief introduction to complex numbers can be found in any calculus lecture dealing with those or also in the book [114].

In other words:

Definition 12.28 (A-stability)**.** *Let $\{y_n\}_n$ the sequence of solutions of a difference method for solving the ODE model problem. Then, this method is A-stable if for arbitrary $\lambda \in \mathbb{C}^- = \{\lambda : Re(\lambda) \leq 0\}$ the approximate solutions are bounded (or even contractive) for arbitrary, but fixed, step size k. That is to say:*

$$|y_{n+1}| \leq |y_n| < \infty \quad \text{for } n = 1, 2, 3, \ldots$$

Remark 12.29. *A-stability is attractive since in particular for stiff problems we can compute with arbitrary step sizes k and do not need any step size restriction.*

Proposition 12.30. *The explicit Euler scheme cannot be A-stable.*

Proof. For the forward Euler scheme, it is $R(z) = 1 + z$. For $|z| \to \infty$ is holds $R(z) \to \infty$ which is a violation of the definition of A-stability. □

Remark 12.31. *More generally, explicit schemes can never be A-stable.*

Example 12.32. *We illustrate the previous statements.*

1. *In Proposition 12.26 we have seen that for the forward Euler method it holds:*

$$y_n = R(z)y_{n-1},$$

where $R(z) = 1 + z$. Thus, according to Definition 12.27 and 12.28, we obtain convergence when the sequence $\{y_n\}$ is contracting:

$$|R(z)| \leq |1 + z| \leq 1. \tag{251}$$

Thus if the value of λ (in $z = \lambda k$) is very big, we must choose a very small time step k in order to achieve $|1 - \lambda k| < 1$. Otherwise the sequence $\{y_n\}_n$ will increase and thus diverge (recall that stability is defined with respect to decreasing parts of functions! Thus, the continuous solution is bounded and consequently the numerical approximation should be bounded, too). In conclusions, the forward Euler scheme is only conditionally stable, i.e., it is stable provided that (251) *is fulfilled.*

2. *For the implicit Euler scheme, we see that a large λ and large k even both help to stabilize the iteration scheme (but be careful, the implicit Euler scheme, stabilizes actually too much. Because it computes contracting sequences also for case where the continuous solution would grow). Thus, no time step restriction is required. Consequently, the implicit Euler scheme is well suited for stiff problems with large parameters/coefficients λ.*

Remark 12.33. *The previous example shows that a careful design of the appropriate discretization scheme requires some work: there is no a priori best scheme. Some schemes require time step size restrictions in case of large coefficients (explicit Euler). On the other hand, the implicit Euler scheme does not need step restrictions but may have in certain cases too much damping. Which scheme should be employed for which problem depends finally on the problem itself and must be carefully thought for each problem again.*

12.3.7.5 Consistency / local discretization error - convergence order We address now the second 'error source' in (237). The consistency determines the precision of the scheme and will finally carry over the local rate of consistency to the global rate of convergence. To determine the consistency we assume sufficient regularity of the exact solution such that we can apply Taylor expansion. The idea is then that all Taylor terms of combined to the discrete scheme. The lowest order remainder term determines finally the local consistency of the scheme.

We briefly formally recall Taylor expansion. For a function $f(x)$ we develop at a point $a \neq x$ the Taylor series:

$$T(f(x)) = \sum_{j=0}^{\infty} \frac{f^{(j)}(a)}{j!}(x - a)^j.$$

Let us continue with the forward Euler scheme and let us now specify the truncation error η_{n-1} in (241):

$$y'(t_{n-1}) + \eta_{n-1} = \frac{y(t_n) - y(t_{n-1})}{k}.$$

To this end, we need information about the solution at the old time step t^{n-1} in order to eliminate $y(t_n)$. Thus we use Taylor and develop $y(t^n)$ at the time point t^{n-1}:

$$y(t^n) = y(t^{n-1}) + y'(t^{n-1})k + \frac{1}{2}y''(\tau^{n-1})k^2$$

We obtain the difference quotient of forward Euler by the following manipulation:

$$\frac{y(t^n) - y(t^{n-1})}{k} = y'(t^{n-1}) + \frac{1}{2}y''(\tau^{n-1})k.$$

We observe that the first terms correspond to (240). Thus the remainder term is

$$\frac{1}{2}y''(\tau^{n-1})k$$

and therefore the truncation error η_{n-1} can be estimated as

$$\|\eta_{n-1}\| \leq \max_{t\in[0,T]} \frac{1}{2}\|y''(t)\|k = O(k).$$

Therefore, the convergence order is k (namely linear convergence speed).

12.3.7.6 Convergence With the help of the two previous subsections, we can easily show the following error estimates:

Theorem 12.34 (Convergence of implicit/explicit Euler)**.** *We have*

$$\max_{t_n\in I} |y_n - y(t_n)| \leq c(T, y)k = O(k),$$

where $k := \max_n k_n$.

Proof. The proof does hold for both schemes, except that when we plug-in the stability estimate one should recall that the backward Euler scheme is unconditionally stable and the forward Euler scheme is only stable when the step size k is sufficiently small. It holds for $1 \leq n \leq N$:

$$\begin{aligned}
|y_n - y(t_n)| = \|e_n\| &= k\|\sum_{k=0}^{n-1} B_E^{n-k}\eta_k\| \\
&\leq k\sum_{k=0}^{n-1} \|B_E^{n-k}\eta_k\| \quad \text{(triangle inequality)} \\
&\leq k\sum_{k=0}^{n-1} \|B_E^{n-k}\|\ \|\eta_k\| \\
&\leq k\sum_{k=0}^{n-1} \|B_E^{n-k}\|\ Ck \quad \text{(consistency)} \\
&\leq k\sum_{k=0}^{n-1} 1\ Ck \quad \text{(stability)} \\
&= kN\ Ck \\
&= T\ Ck, \quad \text{where we used } k = T/N \\
&= C(T, y)k \\
&= O(k)
\end{aligned}$$

□

Theorem 12.35 (Convergence of trapezoidal rule)**.** *We have*

$$\max_{t\in I} |y_n(t) - y(t)| \leq c(T, y)k^2 = O(k^2),$$

The main message is that the Euler schemes both converge with order $O(k)$ (which is very slow) and the trapezoidal rule converges quadratically, i.e., $O(k^2)$.

Let us justify the convergence order for the forward Euler scheme in more detail now.

Theorem 12.36. *Let $I := [0,T]$ the time interval and $f : I \times \mathbb{R}^d \to \mathbb{R}^d$ continuously differentiable and globally Lipschitz-continuous with respect to y:*

$$\|f(t,y) - f(t,z)\|_2 \leq L\|y - z\|_2$$

for all $t \in I$ and $y, z \in \mathbb{R}^d$. Let y be the solution to

$$y' = f(t,y), \quad y(0) = y_0.$$

Furthermore let $y_n, n = 1, \ldots, n$ the approximations obtained with the Euler scheme at the nodal points $t_n \in I$. Then it holds

$$\|y(t_n) - y_n\|_2 \leq \frac{(1+Lk)^n - 1}{2L}\|y''\|\ k \leq \frac{e^{LT} - 1}{2L}\|y''\|\ k = c(T,y)k = O(k),$$

for $n = 0, \ldots, N$.

Proof. The proof follows [70], but consists in working out the steps shown at the beginning of Section 12.3.7.3, Section 12.3.7.4, and Section 12.3.7.5. □

12.3.8 Refinements of A-stability

We extend the terminology of A-stability [106]. Moreover, the previous stability functions shall also be used to represent physical oscillations, which are related to the imaginary unit i. To this end, we have:

1. A-stability:
$$|R(z)| \leq 1, \quad z \leq 0$$
This ensures that the discrete solution remains bounded.

2. Strict A-stability:
$$|R(z)| \leq 1 - ck, \quad z \leq -1.$$
The discrete solution is bounded for inhomogeneous right hand sides or irrugular initial data.

3. Strong A-stability:
$$|R(z)| \leq \kappa < 1, \quad z \leq -1.$$
Damping of high-frequency errors and robust against local errors. See for instance Section 2.7 for an illustration.

4. Numerical dissipation. For physical oscillations, the numerically introduced dissipation should be as small as possible. This means:
$$R(\pm i) \sim 1.$$

It is clear that only implicit schemes can be A-stable. Thus, explicit schemes play only a minor role in solving numerically PDEs. Let us provide more examples. For the implicit Euler scheme, it holds:

$$R(z) = \frac{1}{1-z}$$

and in $-i$:

$$|R(-i)| = \left|\frac{1}{1+i}\right| = \frac{1}{\sqrt{2}} < 1.$$

We nicely see that the damping in the implicit Euler scheme is too strong and will damp out physical oscillations. Therefore, this scheme is not suited for wave equations. Now, let us consider the Crank-Nicolson scheme:

$$\lim_{z\to\infty} \frac{1 + \frac{1}{2}z}{1 - \frac{1}{2}z} = -1,$$

which means A-stability. Also we have for $z = -i$:

$$|R(-i)| = |\frac{1 - \frac{1}{2}i}{1 + \frac{1}{2}i}| = 1.$$

Thus, physical oscillations can be perfectly represented. However, any small disturbances (e.g., even round-off errors), can lead to a blow-up of the solution; see again Section 2.7. One possibility is a k-shift towards the implicit side [100]:

$$0.5 + k$$

This scheme is strictly A-stable and still of second order as proven in [100].

12.3.9 Stiff problems

An essential difficulty in developing stable numerical schemes for temporal discretization using finite differences is associated with **stiffness**, which we shall define in the following. Stiffness is very important in both ODE and (time-dependent) PDE applications. The latter situation will be discussed in Section 12.3.10.

Definition 12.37. *An IVP is called **stiff** (along a solution $y(t)$) if the eigenvalues $\lambda(t)$ of the Jacobian $f'(t, y(t))$ yield the stiffness ratio:*

$$\kappa(t) := \frac{\max_{Re\lambda(t)<0} |Re\lambda(t)|}{\min_{Re\lambda(t)<0} |Re\lambda(t)|} \gg 1.$$

In the above case for the model problem, the eigenvalue corresponds directly to the coefficient λ.

It is however not really true to classify any ODE with large coefficients $|\lambda| \gg 1$ always as a stiff problem. Here, the Lipschitz constant of the problem is already large and thus the discretization error asks for relatively small time steps. Rather stiff problems are characterized as ODE solutions that contain various components with (significant) different evolutions over time; that certainly appears in ODE systems in which we seek $y(t) \in \mathbb{R}^n$ rather than $y(t) \in \mathbb{R}$.

For PDEs, we know that the stiffness matrix of the Laplacian satisfies $K \sim \frac{1}{h^2}$ and therefore it holds for the condition number $\kappa \sim \frac{1}{h^2}$ (see Section 10.1). Thus, problems involving the Laplacian are always stiff.

12.3.10 Numerical analysis: stability analysis

We apply the concepts of the previous sections now to the heat equation. and simply replace y by u. The heat equation is a good example of a stiff problem (for the definition we refer back to Section 12.3.9) and therefore, implicit methods are preferred in order to avoid very, very small time steps in order to obtain stability. We substantiate this claim in the current section.

The model problem is

$$\begin{aligned} \partial_t u - \Delta u = 0 \quad & \text{in } \Omega \times I \\ u = 0 \quad & \text{on } \partial\Omega \times I, \\ u(0) = u^0 \quad & \text{in } \Omega \times \{0\}. \end{aligned}$$

Spatial discretization yields:

$$\begin{aligned} (\partial_t u_h, \phi_h) + (\nabla u_h, \nabla \phi_h) = 0, \\ (u_h^0, \phi_h) = (u^0, \phi_h). \end{aligned}$$

Backward Euler time discretization yields:

$$(u_h^n, \phi_h) + k(\nabla u_h^n, \nabla \phi_h) = (u_h^{n-1}, \phi_h)$$

It holds

Proposition 12.38. *A basic stability estimate for the above problem is:*

$$\|u_h^n\| \leq \|u_h^0\| \leq \|u^0\| \tag{252}$$

Proof. For the stability estimate we proceed as earlier and make a special choice for a test function: $\phi_h := u_h^n$. We then obtain:

$$\|u_h^n\|_{L^2}^2 + k|\nabla u_h^n|_{H^1}^2 = (u_h^{n-1}, u_h^n) \leq \frac{1}{2}\|u_h^{n-1}\|_{L^2}^2 + \frac{1}{2}\|u_h^n\|_{L^2}^2$$

which yields:

$$\frac{1}{2}\|u_h^n\|_{L^2}^2 - \frac{1}{2}\|u_h^{n-1}\|_{L^2}^2 + k|\nabla u_h^n|_{H^1}^2 \leq 0, \quad \forall n = 1, \ldots, N.$$

Of course it further holds:

$$\begin{aligned} & \frac{1}{2}\|u_h^n\|_{L^2}^2 - \frac{1}{2}\|u_h^{n-1}\|_{L^2}^2 \leq 0, \quad \forall n = 1, \ldots, N, \\ \Leftrightarrow \quad & \frac{1}{2}\|u_h^n\|_{L^2}^2 \leq \frac{1}{2}\|u_h^{n-1}\|_{L^2}^2 \end{aligned}$$

Taking the square root and applying the result back to $n = 0$ yields

$$\frac{1}{2}\|u_h^n\|_{L^2} \leq \frac{1}{2}\|u_h^0\|_{L^2}$$

It remains to show the second estimate:

$$\|u_h^0\| \leq \|u^0\|$$

In the initial condition we choose the test function $\varphi := u_h^0$ and obtain:

$$(u_h^0, u_h^0) = (u^0, u_h^0) \leq \frac{1}{2}\|u^0\| + \frac{1}{2}\|u_h^0\|$$

which shows the assertion. □

12.3.10.1 Stability analysis for backward Euler A more detailed stability analysis that brings in the spatial discretization properties is based on the matrix notation and is as follows. We recapitulate and start from (see Section 12.3.6):

$$Mu_h^n + kKu_h^n = Mu_h^{n-1}$$

Then:

$$\begin{aligned} & Mu_h^n + kKu_h^n = Mu_h^{n-1} \\ \Rightarrow\ & u_h^n + kM^{-1}Ku_h^n = u_h^{n-1} \\ \Rightarrow\ & (I + kM^{-1}K)u_h^n = u_h^{n-1} \end{aligned}$$

Compare to the ODE notation:

$$(1 + z)y^n = y^{n-1}$$

Thus, the role of $z = k\lambda$ is taken by $kM^{-1}K$ thus $\lambda \sim M^{-1}K$. We know for the condition numbers

$$\kappa(M) = O(1), \qquad \kappa(K) = O(h^{-2}), \quad h \to 0.$$

We proceed similar to ODEs, and obtain

$$u_h^n = (I + kM^{-1}K)^{-1}u_h^{n-1}$$

The goal is now to show that

$$|(I + kM^{-1}K)^{-1}| \leq 1.$$

To this end, let $\mu_j, j = 1, \ldots, M = dim(V_h)$ be the eigenvalues of the matrix $M^{-1}K$ ranging from the smallest eigenvalue $\mu_1 \sim O(1)$ to the largest eigenvalue $\mu_M \sim O(h^{-2})$; for a proof, we refer the reader to [82][Section 7.7]. Then:

$$|(I + kM^{-1}K)^{-1}| = \max_j \frac{1}{1 + k\mu_j} = \frac{1}{1 + k\mu_M}.$$

To have a stable scheme, we must have:

$$\frac{1}{1+k\mu_M} < 1.$$

We clearly have then again from (252):

$$\|u_h^n\| \leq \|u_h^{n-1}\| \quad \Rightarrow \quad \|u_h^n\| \leq \|u_h^0\|$$

Furthermore, we see that the largest eigenvalue μ_M will increase as $O(h^{-2})$ when h tends to zero. Using the backward Euler or Crank-Nicolson scheme, this will be not a problem and both schemes are **unconditionally stable.**

12.3.10.2 Stability analysis for explicit Euler Here, the stability condition reads:

$$u_h^n = (I - kM^{-1}K)u_h^{n-1}$$

yielding

$$|(I - kM^{-1}K)^{-1}| = |\max_j(1 - k\mu_j)| = |1 - k\mu_M|.$$

As before $\mu_M = O(h^{-2})$. To establish a stability estimate of the form $\|u_h^n\| \leq \|u_h^{n-1}\|$, it is required that

$$|1 - k\mu_M| \leq 1.$$

Resolving this estimate, we obtain

$$k\mu_M \leq 2 \quad \Leftrightarrow \quad k \leq \frac{2}{\mu_M} = O(h^2),$$

thus

$$k \leq Ch^2, \quad C > 0. \tag{253}$$

Here, we only have **conditional stability in time**, namely the time step size k depends on the spatial discretization parameter h. Recall that in terms of accuarcy and discretization errors we want to work with small h, then the time step size has to be extremely small in order to have a stable scheme. In practice this is in most cases not attrative at all and for this reason the forward Euler scheme does not play a role despite that the actual solution is cheaper than solving an implicit scheme.

Exercise 10. *Derive stability estimates for the Crank-Nicolson scheme.*

12.3.11 Convergence results

We follow [99]. To this end, we consider a slighly more general problem statement than in the previous sections of this chapter.

Formulation 12.39. *Let $\Omega \subset \mathbb{R}^n$ be a bounded domain with $n = 1, 2, 3$ and all given data sufficiently smooth. Find $u : \Omega \times (0, \infty) \to \mathbb{R}$ such that*

$$\begin{aligned} \partial_t u + A(u) &= f \quad \text{in } \Omega \times (0, \infty) \\ u(0) &= u_0 \quad \text{in } \Omega \times \{0\} \\ u(x,t) &= u_D \quad \text{on } \partial\Omega_D \times (0, \infty) \\ \alpha\partial_n u + \delta u &= u_R \quad \text{on } \partial\Omega_R \times (0, \infty) \end{aligned}$$

with the operator (see also Section 4.3.2)

$$A = -\nabla \cdot (\alpha\nabla) + \beta \cdot \nabla + \gamma.$$

Then, discretize this problem in space with a Galerkin FEM method of order $r \geq 2$. Be careful, linear FEM have the order $r = 2$ (not starting with 1!).

Theorem 12.40 (Convergence order; e.g., [93]; summarized in [99])**.** *Let all given data be sufficiently regular and the spatial discretization appropriate. Let the orders for spatial and temporal discretization be $r \geq 2$ and $s \geq 1$, respectively. Then, we have optimal convergence for u_{kh}^n to $u(t_n)$ for $k, h \to 0$, uniformely on bounded time intervals, i.e.,*

$$\max_{0 \leq t_n \leq T} \|u(t_n) - u_{kh}^n\|_{L^2(\Omega)} = O(h^r + k^s).$$

In the case of irregular data, the convergence rate may be reduced to $o(1)$ in the worst case.

Remark 12.41. *In several papers in the '80s, Rannacher has proven how the optimal convergence order can be retained in the case of rough or irrgular initial data: [99, 100].*

Remark 12.42. *Some simple convergence analysis computations to show computationally the convergence order are performed in Section 12.3.12.1.*

12.3.12 Numerical tests

The series of numerical tests in this section has three goals:

- to show some figures to get an impression about simulation results;
- to show computationally that the stability results are indeed satisfied or violated;
- to show computationally satisfaction of the (discrete) maximum principle.

We consider the heat equation

$$\begin{aligned} \partial_t u - \Delta u &= 0, \quad \text{in } \Omega \times I \\ u &= 0 \quad \text{on } \partial\Omega \times I, \\ u(x,0) &= \sin(x)\sin(y) \quad \text{in } \Omega \times \{0\}. \end{aligned}$$

We compute 5 time steps, i.e., $T = 5s$, with time step size $k = 1s$. The computational domain is $\Omega = (0,\pi)^2$. We use a One-Step-θ scheme with $\theta = 0$ (forward Euler) and $\theta = 1$ (backward Euler). The full programming code can be found in Section 17.2.8.

The graphical results are displayed in the Figures 46 - 49. Due to stability reasons and violation of the condition $k \leq ch^2$, the forward Euler scheme is unstable (Figures 48 - 49). By reducing the time step size to $k = 0.01s$ (the critical time step value could have been computed - the value $k = 0.01s$ has been found by trial and error simply), we obtain stable results for the forward Euler scheme. These findings are displayed in Figure 50. However, to reach $T = 5s$, we need to compute 500 time steps, which is finally more expensive, despite being an explicit scheme, than the implicit Euler method.

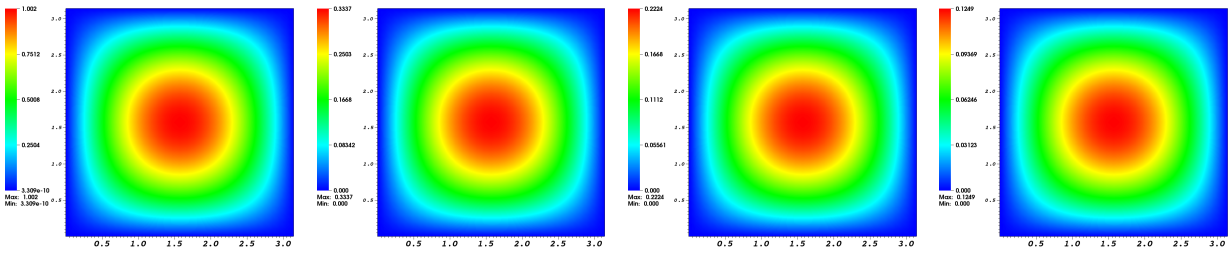


Figure 46: Heat equation with $\theta = 1$ (backward Euler) at $T = 0, 1, 2, 5$. The solution is stable and satisfies the parabolic maximum principle. The color scale is adapted at each time to the minimum and maximum values of the solution.

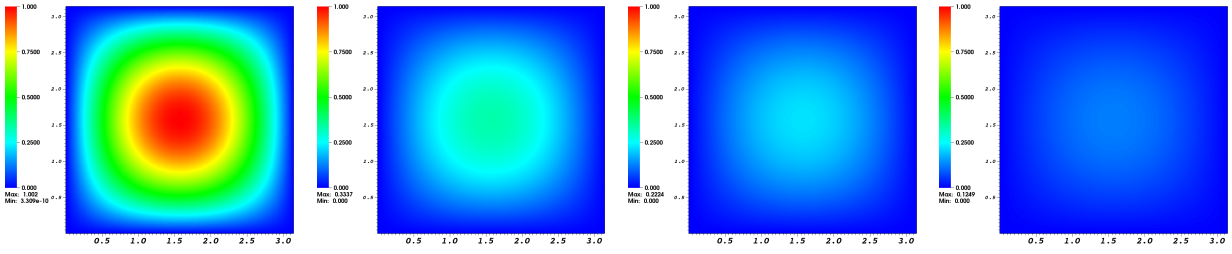


Figure 47: Heat equation with $\theta = 1$ (backward Euler) at $T = 0, 1, 2, 5$. The solution is stable and satisfies the parabolic maximum principle. The color scale is fixed between 0 and 1.

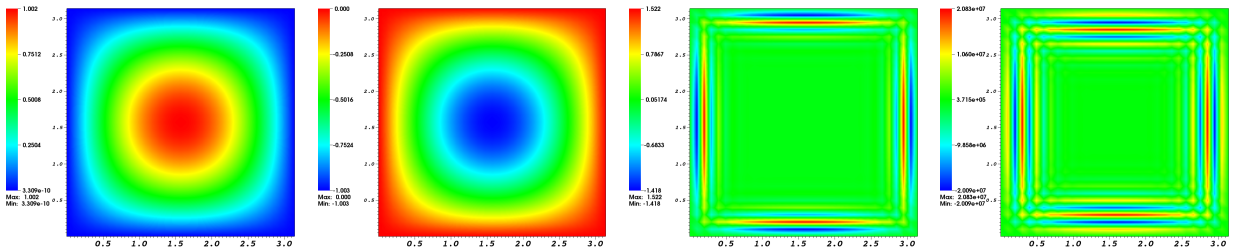


Figure 48: Heat equation with $\theta = 0$ (forward Euler) at $T = 0, 1, 2, 5$. The solution is unstable, showing non-physical oscillations, because the time step restriction (253) is violated. The color scale is adapted at each time to the minimum and maximum values of the solution.

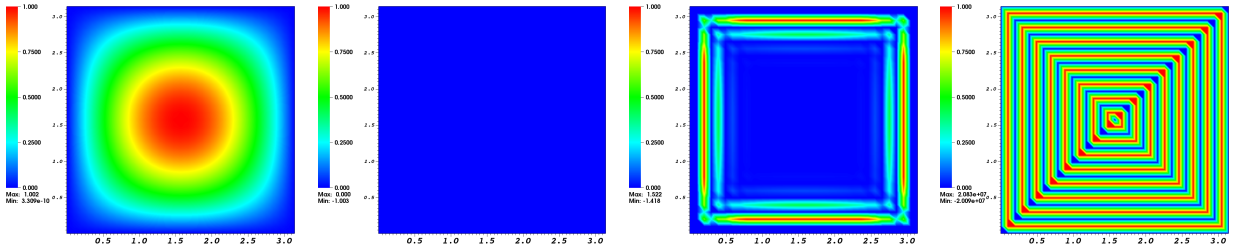


Figure 49: Heat equation with $\theta = 0$ (forward Euler) at $T = 0, 1, 2, 5$. The solution is unstable, showing non-physical oscillations, because the time step restriction (253) is violated. The color scale is fixed between 0 and 1.

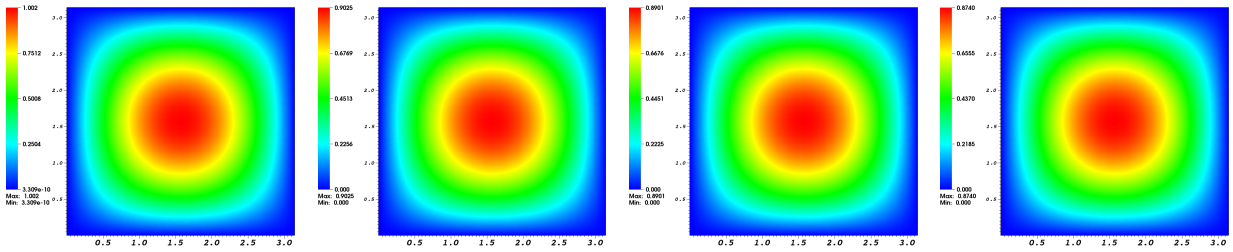


Figure 50: Heat equation with $\theta = 0$ at $T = 0, 1, 2, 5$ and time step size $k = 0.01s$. Here the results are stable since the time step size k has been chosen sufficiently small to satisfy the stability estimate (253). Of course, to obtain results at the same times T as before, we now need to compute more solutions, i.e., $N = 500$, rather than 5 as in the other tests. The color scale is adapted at each time to the minimum and maximum values of the solution.

12.3.12.1 Computational convergence analysis With the help of Chapter 16, we perform a brief temporal convergence analysis for $\theta = 1$ and $\theta = 0.5$. In space, the mesh is five times refined. As time step sizes, we choose $k = 1, 0.5, 0.25$. As goal functional, we choose the spatial L^2 norm at the end time value $T = 25s$:

$$J(u_{kh}(T)) = \|u_{kh}(T)\|_{L^2(\Omega)}.$$

Then, for the backward Euler scheme, we obtain:

```
alpha = 1/log(2) * log((6.644876e-02 - 1.236463e-01) /(1.236463e-01 - 2.485528e-01) )
ans = abs(alpha) = 1.1268
```

and for the Crank-Nicolson scheme:

```
alpha = 1/log(2) * log ((2.352296e-05 - 4.243702e-02 ) - (4.243702e-02 - 1.660507e-01))
ans = abs(alpha) = 3.6224
```

For the latter, the value is better than expected, which might be due that the spatial mesh is still too course. See the hints in Section 16.3.

12.3.13 Student project in the FEM-C++ course in the summer 2018

We consider another numerical example with the heat equation inspired from control engineering. This test and the corresponding C++ code were designed from scratch in the FEM-C++ class in the summer 2018 at LUH by J. Goldenstein and J. Grashorn [60]. The programming code was written from scratch and the graphical output was done with MATLAB using the MEX functionality. Specifically, the C++ code delivers `csv` files that can be read by MATLAB.

We consider the following heat equation. Let $\Omega \subset \mathbb{R}^2$ and find $u = u(x,t) : \Omega \times I \to \mathbb{R}$:

$$\begin{aligned} \partial_t u - \operatorname{div}(\alpha \nabla u) &= f_s(x,t), && \text{in } \Omega \times I, \\ u &= 0 && \text{on } \partial\Omega \times I, \\ u(x,0) &= 0 && \text{in } \Omega \times \{0\}. \end{aligned}$$

Here α describes the heat conductivity and f_s the heat energy. In the numerical tests, $\alpha = 10$ is chosen.

Besides code development, one main goal of the simulations was to compare again different time stepping schemes with respect to their stability:

- $\theta = 0$: explicit Euler (simulation results displated in Figure 53;
- $\theta = 0.5 + k$: shifted Crank-Nicolson (simulation results displated in Figure 54.

The initial heat sources were described as sketched in Figure 51 [left].

In particular the right hand side $f_s(x,t)$ is itself determined by solving another ordinary differential equation for the heat energy:

$$\partial_t \Theta = \Theta_0 + \omega(\Theta_{H,C} - \Theta(t)),$$

with the evolution shown in Figure 51 [right]. Here $\omega = 0.6$. We set:

$$f_s(x,t) = \Theta(t).$$

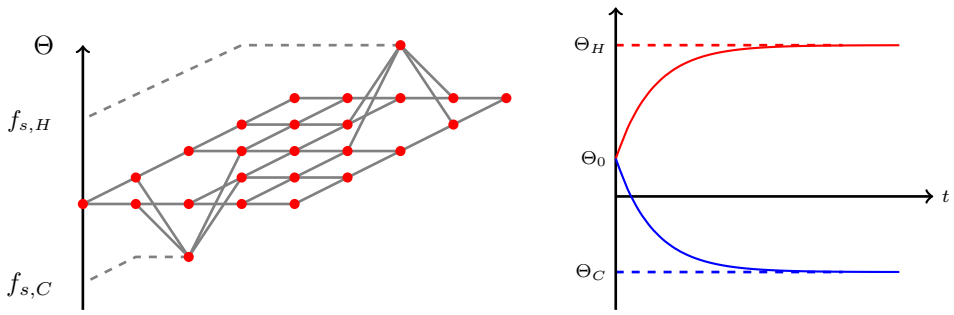


Figure 51: Left: heat sources $f_s(x,t) := \Theta(t)$ in the domain Ω. At right, the evolution of $\Theta(t)$ is shown. Figures provided by [60].

The parameters for the simulations are: the time step size $k = 0.1s$, the end time value $T_{max} = 5$, and furthermore $\Theta_0 = 0$ and $\Theta_H = 1$ and $\Theta_C = -1$. In the spatial discretization 20 nodes in each direction are chosen. The resulting force data $f_s(x,t)$ are plotted in Figure 52 and the simulations results of solving the final heat equation are displayed in Figure 53 and Figure 54.

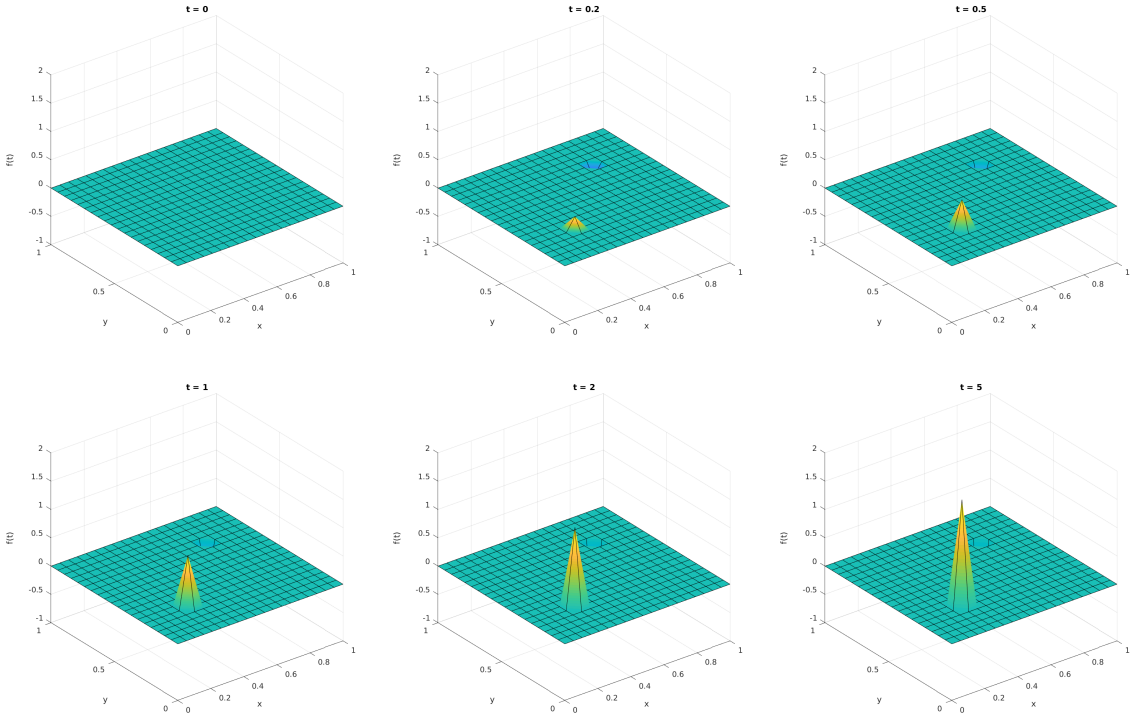


Figure 52: Evolution of $f_s(x,t)$. Figures provided by [60].

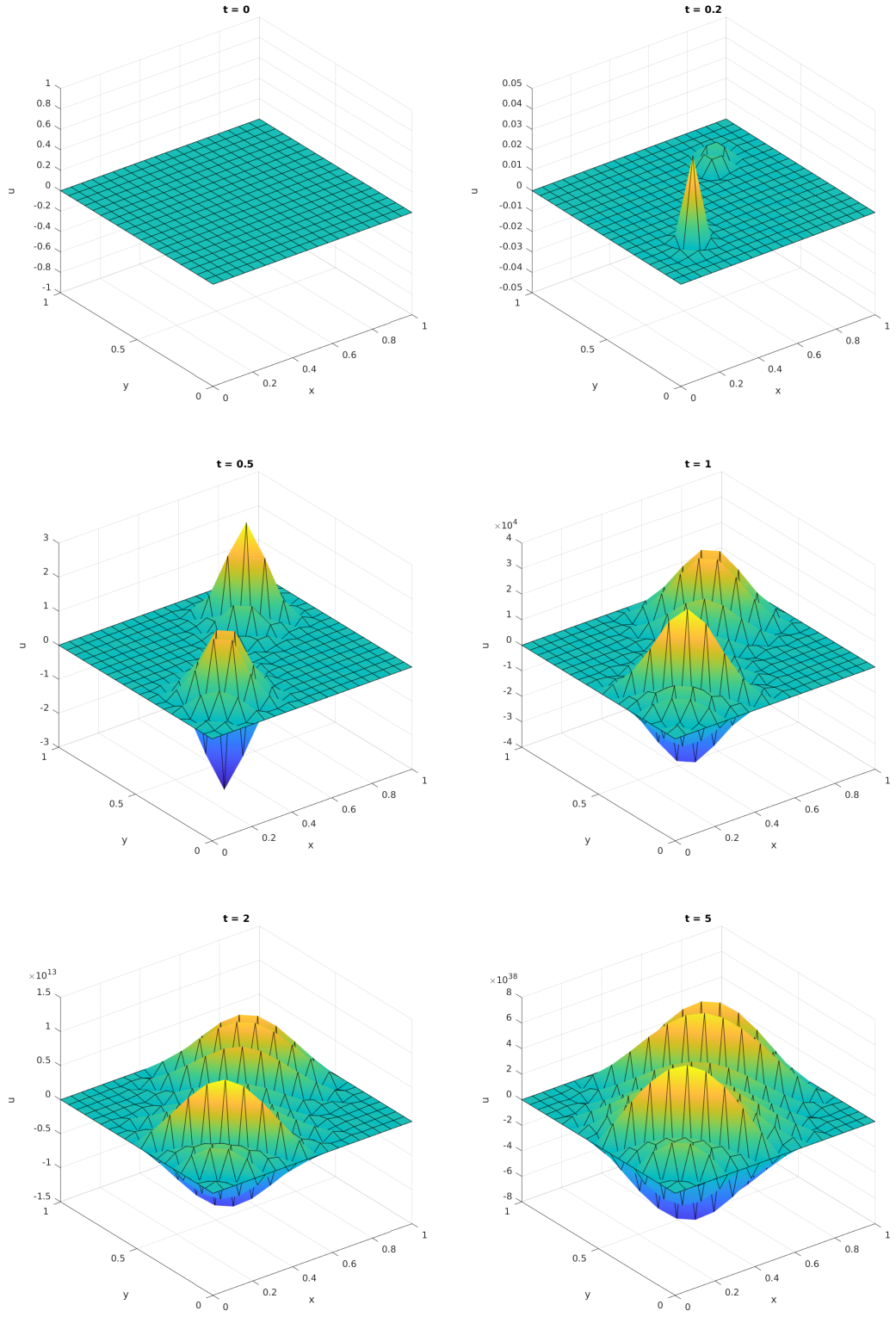


Figure 53: Numerical simulations using the explicit Euler scheme. Solution plots at $t = 0, 0.2, 0.5, 1, 2, 5$ are displayed. We observe blow-up since the CFL-condition $k \leq ch^2$ for numerical stability in time is violated. Figures provided by [60].

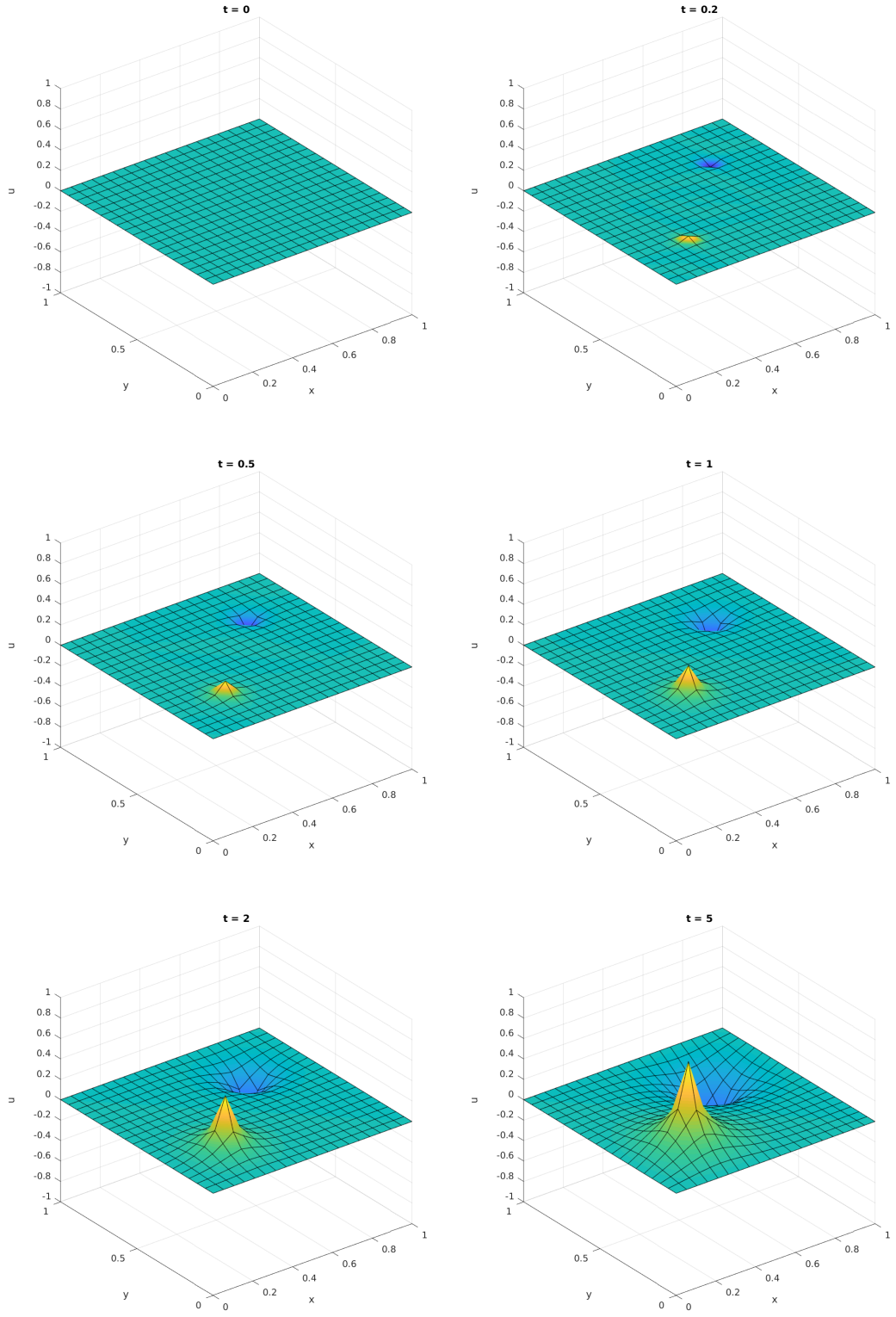


Figure 54: Numerical simulations using the shifted Crank-Nicolson scheme. Solution plots at $t = 0, 0.2, 0.5, 1, 2, 5$ are displayed. Here, we observe stable results. Figures provided by [60].

12.4 Time discretization by operator splitting

Originally, time discretization schemes with operator splitting where introduced mainly for the Navier-Stokes equations in order to decouple difficulties associated to the nonlinearity (the convection term) and the incompressiblity condition. However, also for other types of problems, the Fractional-Step-θ scheme, the most prominent technique, appeared also to be very advantageous. The most striking feature is it little numerical dissipation and being strongly A-stable such that we have the full smoothing property.

12.4.1 Derivation

Let V be a real Hilbert space and let $A : V \to V$. We consider as model problem the initial value problem:

$$\partial_t u + A(u) = f, \qquad u(0) = u_0.$$

In the following we split A into two operators A_1 and A_2 such that

$$A = A_1 + A_2$$

As before let k the time step size.

12.4.2 Peaceman-Rachford time discretization

12.4.2.1 Basic scheme A very old and famous scheme is the Peaceman-Rachford scheme:

Formulation 12.43 (Peaceman-Rachford time discretization)**.** *Let $u^0 = u_0$ be the discrete initial value at $n = 0$. For $n \geq 0$, let u^n be the previous time step solution. We compute u^{n+1} in a two-step procedure via the intermediate value $u^{n+1/2}$:*

$$\frac{u^{n+1/2} - u^n}{2k} + A_1(u^{n+1/2}) + A_2(u^n) = f^{n+1/2},$$
$$\frac{u^{n+1} - u^{n+1/2}}{2k} + A_1(u^{n+1/2}) + A_2(u^{n+1}) = f^{n+1},$$

where $u^{n+\alpha} := u(t^{n+\alpha})$ and $f^{n+\alpha} := f(t^{n+\alpha})$.

12.4.2.2 Numerical analysis: stability and convergence We follow [29], but also refer to the references cited therein.

Let us work in $\mathbb{R}^n$. Let $f = 0$ and $u_0 \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n\times n}$ is symmetric and positive definite. Furthermore

$$A_1 = \alpha A, \quad A_2 = \beta A$$

with $\alpha + \beta = 1$ and $0 < \alpha, \beta < 1$.

From ODEs it is well-known that the matrix exponential function is a solution to the above model problem:

$$u(t) = \exp(-tA)u_0.$$

12.4.2.2.1 Stability The goal is now to show a stability estimate as

$$u^{n+1} = R(z)u^n$$

with $|R(z)| \leq 1$ similar to Section 12.3.7.4.

To derive $R(z)$ for a splitted scheme, we combine the two equations into one single equation and then proceed as in Section 12.3.7.4. Using the splitting à la Peaceman-Rachford and the definitions of A_1 and A_2, we obtain:

$$u^{n+1} = (I + \beta\frac{k}{2}A)^{-1}(I - \alpha\frac{k}{2}A)(I + \alpha\frac{k}{2}A)^{-1}(I - \beta\frac{k}{2}A)u^n.$$

Since A is symmetric positive definite, there exists a basis of eigenvectors. Consequently it holds

$$u_i^{n+1} = (1+\beta\frac{k}{2}\lambda_i)^{-1}(1-\alpha\frac{k}{2}\lambda_i)(1+\alpha\frac{k}{2}\lambda_i)^{-1}(1-\beta\frac{k}{2}\lambda_i)u_i^n,$$

where $\lambda_i > 0$ for $i = 1,\dots,N$ denoting the ith eigenvalue of the matrix A. We assume that the eigenvalues are ordered as

$$\lambda_1 \le \lambda_2 \le \ldots \le \lambda_N.$$

In order to analyze further, let us consider the rational function

$$R(z) = \frac{(1-\frac{\alpha}{2}x)(1-\frac{\beta}{2}x)}{(1+\frac{\alpha}{2}x)(1+\frac{\beta}{2}x)}$$

Careful inspection yields that this stability function shares on the first view some similarities with the Crank-Nicolson scheme. Indeed the Peaceman-Rachford scheme is inconditionally stable when

$$|R(z)| < 1 \quad \text{for all} z > 0,$$

which is the case for the rational function under consideration. However, we have

$$\lim_{z\to\infty} |R(z)| = 1$$

and therefore the scheme is only A-stable but not strongly A-stable. For stiff problems with $\lambda_N/\lambda_1 \gg 1$ the different components associated to small and large eigenvalues of A are not simultaneously damped. And the scheme may possibly suffer from irregular initial data or numerical noice for long-term simulations.

12.4.2.2.2 Consistency and convergence To derive the order of the scheme, we develop the analytical solution $\exp(-tA)$ via Taylor. Let us do this for $z = tA$:

$$e^{-z} = 1 - z + \frac{1}{2}z^2 + z^2 b(z).$$

On the other hand, the rational function can be re-written as:

$$R(z) = 1 - z + \frac{1}{2}z^2 + z^2 c(z).$$

For the higher-order terms it holds

$$\lim_{z\to 0} b(z) = \lim_{z\to 0} c(z) = 0.$$

Since the first three terms coincide up to z^2, the scheme is second-order accurate for the model problem.

The convergence follows with similar arguments as shown in Section 12.3.7.6.

Remark 12.44. *For $\alpha = \beta = \frac{1}{2}$ the two linear systems in the splitting scheme have the same matrix $I + \frac{k}{4}A$.*

12.4.2.3 Application to Navier-Stokes

Formulation 12.45 (Peaceman-Rachford time discretization applied to NSE)**.** *Let $u^0 = u_0$ be the discrete initial value at $n = 0$. For $n \ge 0$, let v^n be the previous time step solution of the velocities. From v^n, we compute $v^{n+1/2}, p^{n+1/2}, v^{n+1}$ via*

$$\textit{Step 1:} \quad \frac{v^{n+1/2}-v^n}{2k} - \frac{\nu}{2}\Delta v^{n+1/2} + \nabla p^{n+1/2} = f^{n+1/2} + \frac{\nu}{2}\Delta v^n - (v^n\cdot\nabla)v^n,$$
$$\nabla\cdot v^{n+1/2} = 0,$$
$$\textit{Step 2:} \quad \frac{v^{n+1}-v^{n+1/2}}{2k} - \frac{\nu}{2}\Delta v^{n+1} + (v^{n+1}\cdot\nabla)v^{n+1} = f^{n+1} + \frac{\nu}{2}\Delta v^{n+1/2} - \nabla p^{n+1/2}.$$

Remark 12.46. *In Step 1, we treat the incompressibility constraint and take the previous convection term time discretization. In Step 2, we then consider the convection term for the current time step. Therefore:*

$$A_1(v) = \nabla\cdot v,$$
$$A_2(v) = (v\cdot\nabla)v.$$

12.4.3 The Fractional-Step-θ scheme

12.4.3.1 Basic scheme The idea is to split the current time interval $[t^n, t^{n+1}]$ into three subintervals. First, we take A_1 implicitly and A_2 explicitly, in the second step, we switch the roles, and the third step is the same as the first.

Formulation 12.47 (Fractional-Step-θ). *Let $\theta \in (0, 1/2)$. Set $u^0 = u_0$ and $n \geq 0$. We compute u^{n+1} in a three-step procedure via the intermediate values $u^{n+\theta}, u^{n+1-\theta}$ as follows:*

$$\begin{aligned}
\frac{u^{n+\theta} - u^n}{\theta k} + A_1(u^{n+\theta}) + A_2(u^n) &= f^{n+\theta},\\
\frac{u^{n+1-\theta} - u^{n+\theta}}{(1-2\theta)k} + A_1(u^{n+\theta}) + A_2(u^{n+1-\theta}) &= f^{n+1-\theta},\\
\frac{u^{n+1} - u^{n+1-\theta}}{\theta k} + A_1(u^{n+1}) + A_2(u^{n+1-\theta}) &= f^{n+1},
\end{aligned}$$

Remark 12.48. *The Fractional-Step-θ scheme is 2nd order for a special choice of θ and strongly A-stable (in contrast to Crank-Nicolson or the Peaceman-Rachford scheme). Consequently, the scheme has the full smoothing property, which is important for rough initial data, rough boundary values or accumulation of other numerical errors (e.g., quadrature) for long-time computations. In addition, the FS-θ scheme contains little numerical dissipation, which is very desirable when dealing with elastodynamics (wave equations) and flow simulations with nonenforced temporal oscillations (e.g., flow benchmarks [118] or FSI benchmarks [77]).*

12.4.3.2 Numerical analysis: stability and convergence Let us now analyse the FS-θ scheme. Again we follow closely the original work [29]. We set $\theta' = 1 - 2\theta$. We have

$$u^{n+1} = (I + \alpha\theta kA)^{-1}(I - \beta\theta kA)(I + \beta\theta' kA)^{-1}(I - \alpha\theta' kA)(I + \alpha\theta kA)^{-1}(I - \beta\theta kA)u^n.$$

12.4.3.2.1 Stability Proceeding as for the Peaceman-Rachford scheme, we use the eigenvalues for each component i:

$$u_i^{n+1} = \frac{(1 - \beta\theta k\lambda_i)^2(1 - \alpha\theta' k\lambda_i)}{(1 + \alpha\theta k\lambda_i)^2(1 + \beta\theta' k\lambda_i)} u_i^n.$$

We consider for simplicity of the presentation again the rational function $R(z)$ defined as

$$R(z) = \frac{(1 - \beta\theta kz)^2(1 - \alpha\theta' kz)}{(1 + \alpha\theta kz)^2(1 + \beta\theta' kz)}$$

We have in particular

$$\lim_{z\to\infty} |R(z)| = \frac{\beta}{\alpha}$$

such that $\alpha \geq \beta$ in order to have unconditional stability for the largest eigenvalues of A.

12.4.3.2.2 Consistency We know for the model problem:

$$e^{-z} = 1 - z + \frac{1}{2}z^2 + z^2 b(z).$$

Thus, we have to see until the FS-θ can match. First, we re-write $R(z)$ such that

$$R(z) = 1 - z + \frac{1}{2}z^2\left(1 + (\beta^2 - \alpha^2)(2\theta^2 - 4\theta + 1)\right) + z^2 c(z).$$

The scheme is first-order accurate when

$$(\beta^2 - \alpha^2) \neq 0 \quad \text{and} \quad (2\theta^2 - 4\theta + 1) \neq 0.$$

The scheme is second-order accurate when

$$(\beta^2 - \alpha^2) = 0 \quad \text{or} \quad (2\theta^2 - 4\theta + 1) = 0.$$

To have second-order accuracy we need specifically:

$$\alpha = \beta$$

Because of $\alpha + \beta = 1$ it then follows $\alpha = \beta = \frac{1}{2}$. Or we need

$$\theta = 1 - \frac{\sqrt{2}}{2}.$$

The question is whether both choices are admissible? Yes, they are, but for $\alpha = \beta = \frac{1}{2}$ we have from

$$\lim_{z\to\infty} |R(z)| = \frac{\beta}{\alpha} = 1$$

Therefore, this choice is not strongly A-stable.

In order to check the second choice we need to express θ in terms of α and β. We begin with the idea that the matrices should be the same for all three substeps of the FS-θ scheme. This means:

$$\alpha\theta = \beta(1 - 2\theta).$$

It can be inferred that

$$\alpha = \frac{1 - 2\theta}{1 - \theta}, \quad \beta = \frac{\theta}{1 - \theta}$$

Knowing that $\alpha \geq \beta$ yields

$$0 < \theta < \frac{1}{3}.$$

In fact

$$\theta = \frac{1}{3} \quad \Rightarrow \quad \alpha = \beta = \frac{1}{2}$$

Therefore, this choice is again only a variant of the Peacemen-Rachford scheme. For θ strictly smaller than $\frac{1}{3}$, we have

$$\lim_{z\to\infty} |R(z)| = \frac{\beta}{\alpha} = \frac{\theta}{1 - 2\theta} < 1.$$

For this reason, the scheme is unconditionally stable for this choice of θ and has good smoothing properties for large eigenvalues of A. The most famous choice for θ is

- $\theta = 1 - \sqrt{2}/2$,
- $\alpha = 2 - \sqrt{2}$,
- $\beta = \sqrt{2} - 1$.

12.4.3.3 Application to Navier-Stokes

Formulation 12.49 (Fractional-Step-θ time discretization applied to NSE)**.** *Let $u^0 = u_0$ be the discrete initial value at $n = 0$. For $n \geq 0$, let v^n be the previous time step solution of the velocities. From v^n, we solve*

$$\textit{Step 1:} \quad \frac{v^{n+\theta} - v^n}{\theta k} - \alpha\nu\Delta v^{n+1/2} + \nabla p^{n+\theta} = f^{n+\theta} + \beta\nu\Delta v^n - (v^n \cdot \nabla)v^n,$$
$$\nabla \cdot v^{n+\theta} = 0,$$

$$\textit{Step 2:} \quad \frac{v^{n+1-\theta} - v^{n+\theta}}{(1 - 2\theta)k} - \beta\nu\Delta v^{n+1-\theta} + (v^{n+1-\theta} \cdot \nabla)v^{n+1-\theta} = f^{n+1-\theta} + \alpha\nu\Delta v^{n+\theta} - \nabla p^{n+\theta},$$

$$\textit{Step 3:} \quad \frac{v^{n+1} - v^{n+1-\theta}}{\theta k} - \alpha\nu\Delta v^{n+1} + \nabla p^{n+1} = f^{n+1} + \beta\nu\Delta v^{n+1-\theta} - (v^{n+1-\theta} \cdot \nabla)v^{n+1-\theta},$$
$$\nabla \cdot v^{n+1} = 0.$$

12.4.4 Modified Fractional-Step-θ schemes

We present two recently proposed modifications of the original FS-θ scheme. Here, the second step is replaced by an extrapolation rather than solving another system.

12.4.4.1 Basic scheme The macro time step is $k = t^{n+1} - t^n$. Let $\theta = 1 - \frac{1}{\sqrt{2}}$ and the initial value $u^0 = u_0$, $n \geq 0$ and u^n is the previous time step solution. Then, we solve:

$$\frac{u^{n+\theta} - u^n}{\theta k} = f(u^{n+\theta}, t^{n+\theta}), \tag{254}$$

$$u^{n+1-\theta} = \frac{1-\theta}{\theta} u^{n+\theta} + \frac{2\theta - 1}{\theta} u^n \frac{u^{n+1} - u^{n+1-\theta}}{\theta k} = f(u^{n+1}, t^{n+1}). \tag{255}$$

Remark 12.50. *This scheme is fully implicit, strongly A-stable, 2nd-order accurate.*

12.4.4.2 Application to Navier-Stokes (I)

Formulation 12.51.

$$\textit{Step 1:} \quad \frac{v^{n+\theta} - v^n}{\theta k} + [-\nu \Delta v^{n+\theta} + v^{n+\theta} \cdot \nabla v^{n+\theta}] + \nabla p^{n+\theta} = f^{n+\theta}, \tag{256}$$

$$\nabla \cdot v^{n+\theta} = 0, \tag{257}$$

$$\textit{Step 2:} \quad v^{n+1-\theta} = \frac{1-\theta}{\theta} v^{n+\theta} + \frac{2\theta - 1}{\theta} v^n, \tag{258}$$

$$\textit{Step 3:} \quad \frac{v^{n+1} - v^{n+1-\theta}}{\theta k} + [-\nu \Delta v^{n+1} + v^{n+1} \cdot \nabla v^{n+1}] + \nabla \tilde{p}^{n+1} = f^{n+1}, \tag{259}$$

$$\nabla \cdot v^{n+1} = 0, \tag{260}$$

$$\textit{Step 4:} \quad p^{n+1} = (1-\theta) p^{n+\theta} + \theta \tilde{p}^{n+1}. \tag{261}$$

12.4.4.3 Application to Navier-Stokes (II)

Formulation 12.52.

$$\textit{Step 1:} \quad \frac{v^{n+\theta} - v^n}{\theta k} + [-\nu \Delta v^{n+\theta} + v^{n+\theta} \cdot \nabla v^{n+\theta}] + \nabla p^{n+\theta} = f^{n+\theta}, \tag{262}$$

$$\nabla \cdot v^{n+\theta} = 0, \tag{263}$$

$$\textit{Step 2:} \quad v^{n+1-\theta} = \frac{1-\theta}{\theta} v^{n+\theta} + \frac{2\theta - 1}{\theta} v^n, \tag{264}$$

$$\textit{Step 3:} \quad \frac{v^{n+1} - v^{n+1-\theta}}{\theta k} + [-\nu \Delta v^{n+1} + v^{n+1} \cdot \nabla v^{n+1}] + \nabla p^{n+1} = f^{n+1}, \tag{265}$$

$$\nabla \cdot v^{n+1} = 0. \tag{266}$$

Remark 12.53. *Here we simply take the last pressure solution and do not work with a tilde pressure.*

12.5 The discontinuous Galerkin (DG) method for temporal discretization

We now introduct a DG discretization in time. The way is very similar to the class Numerik 2 [117].

Definition 12.54. *Let $q \in \mathbb{N}$ and $I = (0,T)$. Then, we define the space*

$$X_{hk} = \{v : I \to V_h : v|_{I_n} \in P_q(I_n), n = 1, \dots, N\}$$

where

$$P_q(I_n) = \{v : I_n \to V_h : v(t) = \sum_{i=0}^{q} v_i t^i,\ v_i \in V_h\}.$$

In words: the space X_{hk} is the space of functions on the time interval I with values in the FE space V_h. On each time interval I_n the functions vary as polynomials of degree at most q.

We recall from Numerik 2 [117] that the functions $v \in X_{hk}$ may be discontinuous in time at the discrete time levels t_n. We use the standard notation

$$v_+^n = \lim_{\varepsilon \to 0^+} v(t_n + \varepsilon), \quad v_-^n = \lim_{\varepsilon \to 0^-} v(t_n + \varepsilon), \quad [v^n] = v_+^n - v_n^n$$

The third definition is the jump of v at t_n. We formulate the DG method for the heat equation as follows:

Formulation 12.55. *Find $u_{hk} \in X_{hk}$ such that*

$$\bar{a}(u_{hk}, \varphi_{hk}) = \bar{l}(\varphi_{hk}) \quad \forall \varphi_{hk} \in X_{hk}.$$

with

$$\bar{a}(u_{hk}, \varphi_{hk}) = \sum_{n=1}^{N} \int_{I_n} ((\partial_t u_{hk}, \varphi_{hk}) + a(u_{hk}, \varphi_{hk}))\, dt + \sum_{n=2}^{N} ([u_{hk}^{n-1}], \varphi_+^{n-1}) + (u_{hk}^0, \varphi_+^0)$$

and the right hand side functional

$$\bar{l}(\varphi_{hk}) = \int_I (f, \varphi_{hk})\, dt + (u^0, \varphi_+^0).$$

Because $v \in X_{hk}$ varies independently on each subinterval I_n, the previous formulation decouples and we obtain:

Formulation 12.56. *For $n = 1, \dots, N$ given $u_{hk,-}^{n-1}$, find $u_{hk} = u_{hk}|_{I_n} \in P_q(I_n)$ such that*

$$\int_{I_n} ((\partial_t u_{hk}, \varphi) + a(u_{hk}, \varphi))\, dt + (u_{hk,+}^{n-1}, \varphi_+^{n-1}) = \int_{I_n} (f, \varphi)\, dt + (u_{hk,-}^{n-1}, \varphi_+^{n-1}) \quad \varphi \in P_q(I_n)$$

where $u_{hk,-}^0 = u^0$.

Using specific values for q, we obtain concrete time-stepping schemes.

Formulation 12.57 (DG(0))**.** *For $q = 0$ with $u_{hk}^n = u_{hk,-}^n = u_{hk,+}^{n-1}$ we obtain for $n = 1, \dots, N$: Find $u_{hk}^n \in V_h$ such that*

$$(u_{hk}^n - u_{hk}^{n-1}, \varphi) + k_n a(u_{hk}^n, \varphi) = \int_{I_n} (f, \varphi)\, dt \quad \forall \varphi \in V_h,\ n = 1, \dots, N$$

with $u_{hk}^0 = u^0$. This scheme is close to the backward Euler scheme. The only difference is in the right hand side, which is here an average of I_n rather than the evaluation of f at t_n.

Formulation 12.58 (DG(1))**.** *For $q = 1$, we obtain a system of equations. First, we define*

$$u_{hk}(t) = v_0 + \frac{t - t_{n-1}}{k_n} v_1, \quad t \in I_n, \quad v_i \in V_h.$$

Then, we obtain:

$$(v_0, \varphi) + k_n a(v_0, \varphi) + (v_1, \varphi) + \frac{1}{2} k_n a(v_1, \varphi) = (u_{hk,-}^{n-1}, \varphi) + \int_{I_n} (f(s), \varphi)\, ds, \quad \forall \varphi \in V_h$$

$$\frac{1}{2} k_n a(v_0, \varphi) + \frac{1}{2}(v_1, \varphi) + \frac{1}{3} k_n a(v_1, \varphi) = \frac{1}{k_n} \int_{I_n} (s - t_{n-1})(f(s), \varphi)\, ds, \quad \forall \varphi \in V_h.$$

12.6 Methods for second-order-in-time hyperbolic problems

For second-order (in time) hyperbolic problems (prototype is the wave equation) the situation is a bit more tricky since we have a second-order in time derivative.

12.6.1 Problem statement

The equation is given by; see also Section 4.3:

$$\begin{aligned} \rho\partial_{tt}^2 u - \nabla\cdot(\nabla u) &= f \quad \text{in } \Omega\times I,\\ u &= 0 \quad \text{on } \partial\Omega\times I,\\ u(x,0) &= u_0(x) \quad \text{in } \Omega\times\{0\},\\ \partial_t u(x,0) &= v_0(x) \quad \text{in } \Omega\times\{0\} \end{aligned}$$

A novel feature in comparison to parabolic problems from the previous sections is **energy conservation**. So far we were concerned with

- Stability;
- Convergence.

Now we also need

- Energy conservation.

Consequently, we need time-discretizations that have the boundary of their stability region on the imaginary axis in order to conserve the energy on the time-discrete level; see Section 12.3.8. As we will see, this further reduces the choices of 'good' time-stepping schemes. The wave equation is important in many applications and for this reason a zoo of time-stepping schemes (the most prominent being the Newmark scheme) has been proposed. Finite difference schemes discretizing directly the second-order time derivative have some problems in the energy conservation. On the other hand, discretizing the wave equation directly with a One-Step-θ scheme is not possible because of the second-order time derivative. Therefore, a common procedure is to re-write the equation into a first-order mixed system. We introduce a second solution variable (the velocity), which also needs to be discretized in space and results in a higher computational cost.

12.6.2 Variational formulations

A formal derivation of a variational form reads:

$$(\partial_{tt}^2 u,\phi) + (\nabla u,\nabla\phi) = (f,\phi) \quad \forall\phi\in H_0^1.$$

From this variational formulation, we seek

$$u\in L^2(I,H_0^1),\quad \frac{du}{dt}\in L^2(I,L^2),\quad \frac{d^2u}{dt^2}\in L^2(I,H^{-1}).$$

Proposition 12.59. *Let $f\in L^2(I,L^2)$ and the initial data $u_0\in H_0^1$ and $v_0\in L^2$ be given. Then the variational problem has a unique solution*

$$u\in L^2(I,H_0^1),\quad \frac{du}{dt}\in L^2(I,L^2).$$

Moreover, the mapping

$$\{f,u_0,v_0\}\to\{u,v\},\quad v=\frac{du}{dt}$$

from

$$L^2(I,L^2)\times H_0^1\times L^2 \quad\to\quad L^2(I,H_0^1)\times L^2(I,H^{-1})$$

is linear and continuous.

Proof. See Wloka [145]. □

We also state the promised first-order mixed system. We recall

$$\rho\partial_{tt}^2 u - \nabla\cdot(\nabla u) = f$$

yielding

$$\begin{aligned}\partial_t u &= v,\\ \rho\partial_t v - \nabla\cdot(\nabla u) &= f.\end{aligned}$$

Then we formulate the variational system:

Formulation 12.60. *Find $(u,v)\in L^2(I,H_0^1)\times L^2(I,L^2)$ with $u_0\in H_0^1$ and $v_0\in L^2$ such that*

$$\begin{aligned}(\partial_t u,\psi) &= (v,\psi) \quad \forall\psi\in L^2,\\ (\rho\partial_t v,\phi)+(\nabla u,\nabla\phi) &= (f,\phi) \quad \forall\phi\in H_0^1.\end{aligned}$$

12.6.3 A space-time formulation

We use the previous formulations to derive space-time variational formulations for the wave equation. We briefly introduce the concept in this section.

Let us consider the second-order hyperbolic PDE:

Formulation 12.61. *Let $\Omega\subset\mathbb{R}^n$ be open and let $I:=(0,T)$ with $T>0$. Find $u:\Omega\times I\to\mathbb{R}$ and $\partial_t u:\Omega\times I\to\mathbb{R}$ such that*

$$\begin{aligned}\rho\partial_t^2 u - \nabla\cdot(a\nabla u) &= f \quad \text{in } \Omega\times I,\\ u &= 0 \quad \text{on } \partial\Omega_D\times I,\\ a\partial_n u &= 0 \quad \text{on } \partial\Omega_N\times I,\\ u &= u_0 \quad \text{in } \Omega\times\{0\},\\ v &= v_0 \quad \text{in } \Omega\times\{0\}.\end{aligned}$$

We now prepare the functions spaces requires for the weak formulation. Let us denote L^2 and H^1 as the usual Hilbert spaces and H^{-1} the dual space to H^1. For the initial functions we assume:

- $u_0\in H_0^1(\Omega)^n$;
- $v_0\in L^2(\Omega)^n$.

For the right hand side source term we assume

- $f\in L^2(I,H^{-1}(\Omega))$, where $L^2(\cdot,\cdot)$ is a Bochner space; see Section 12.2. Specifically, we remind the reader that the initial values are a priori not well defined since they are in L-spaces. However, we have Theorem 12.3 ensuring that the initial data are continuous in time and therefore well defined.

We introduce the following short-hand notation:

- $H:=L^2(\Omega)^n$;
- $V:=H_0^1(\Omega)^n$;
- V^* is the dual space to V.
- $\bar{H}:=L^2(I,H)$;
- $\bar{V}:=\{v\in L^2(I,V)|\ \partial_t v\in\bar{H}\}$.

Theorem 12.62. *If the operator $A:=-\nabla\cdot(a\nabla u)$ satisfies the coercivity estimate:*

$$(Au,u)\geq\beta\|u\|_1^2,\quad u\in V,\quad \beta>0,$$

then there exists a unique weak solution with the properties:

- $u \in \bar{V} \cap C(\bar{I}, V)$;
- $\partial_t u \in \bar{H} \cap C(\bar{I}, H)$;
- $\partial_t^2 u \in L^2(I, V^*)$.

Proof. See Lions and Magenes, Lions or Wloka. □

Definition 12.63. *The previous derivations allow us to define a compact space-time function space for the wave equation:*

$$X := \{v \in \bar{V} \,|\, v \in C(\bar{I}, V), \partial_t v \in C(\bar{I}, H), \partial_t^2 v \in L^2(I, V^*)\}.$$

To design a Galerkin time discretization, we need to get rid of the second-order in time derivative and therefore usually the wave equation is re-written in terms of a mixed first-order system:

Formulation 12.64. *Let $\Omega \subset \mathbb{R}^n$ be open and let $I := (0, T)$ with $T > 0$. Find $u : \Omega \times I \to \mathbb{R}$ and $\partial_t u = v : \Omega \times I \to \mathbb{R}$ such that*

$$\begin{aligned}
\rho \partial_t u - \rho v &= 0 \quad \text{in } \Omega \times I, \\
\rho \partial_t v - \nabla \cdot (a \nabla u) &= f \quad \text{in } \Omega \times I, \\
u &= 0 \quad \text{on } \partial\Omega_D \times I, \\
a \partial_n u &= 0 \quad \text{on } \partial\Omega_N \times I, \\
u &= u_0 \quad \text{in } \Omega \times \{0\}, \\
v &= v_0 \quad \text{in } \Omega \times \{0\}.
\end{aligned}$$

To derive a space-time formulation, we first integrate formally in space:

$$\begin{aligned}
A_u(U)(\psi^u) &= (\rho \partial_t u, \psi^u) - (\rho v, \psi^u), \\
A_v(U)(\psi^v) &= (\rho \partial_t v, \psi^v) + (a \nabla u, \nabla \psi^v) - (f, \psi^v)
\end{aligned}$$

And then in time:

$$\begin{aligned}
\bar{A}_u(U)(\psi^u) &= \int_I \left((\rho \partial_t u, \psi^u) - (\rho v, \psi^u) \right) dt + (u(0) - u_0, \psi^u(0)), \\
\bar{A}_v(U)(\psi^v) &= \int_I \left((\rho \partial_t v, \psi^v) + (a \nabla u, \nabla \psi^v) - (f, \psi^v) \right) dt + (v(0) - v_0, \psi^v(0)).
\end{aligned}$$

The total problem reads:

Formulation 12.65. *Find $U = (u, v) \in X_u \times X_v$ with $X_u = X$ and $X_v := \{w \in \bar{H} | w \in C(\bar{I}, H), \partial_t w \in L^2(I, V^*)\}$ such that*

$$\bar{A}(U)(\Psi) = 0 \quad \forall \Psi = (\psi^u, \psi^v) \in X_u \times X_v$$

where

$$\bar{A}(U)(\Psi) := \bar{A}_v(U)(\psi^v) + \bar{A}_u(U)(\psi^u).$$

Such a formulation is the starting point for the discretization: either in space-time or using a sequential time-stepping scheme and for instance FEM in space.

12.6.4 Energy conservation and consequences for good time-stepping schemes

Another feature of the wave equation is energy conservation. In the ideal case, $f = 0, \rho = 1$, and no dissipation, build the variational form and obtain:

$$(\partial_{tt}^2 u, \phi) + (\nabla u, \nabla \phi) = 0.$$

Using $\phi = \partial_t u$ as test function, we obtain:

$$(\partial_{tt}^2 u, \partial_t u) + (\nabla u, \nabla \partial_t u) = 0,$$

which yields:

$$\frac{1}{2}\frac{d}{dt}(\partial_t u, \partial_t u) + \frac{1}{2}\frac{d}{dt}(\nabla u, \nabla u) = 0,$$

which is

$$\frac{1}{2}\frac{d}{dt}\|\partial_t u\|^2 + \frac{1}{2}\frac{d}{dt}\|\nabla u\|^2 = 0.$$

Remark 12.66. *It holds:*

$$\frac{1}{2}\frac{d}{dt}\int_\Omega \partial_t u \cdot \partial_t u \, dt = \frac{1}{2}\int_\Omega \partial_t(\partial_t u \cdot \partial_t u)\, dt = \frac{1}{2}\int_\Omega 2\partial_t^2 u \cdot \partial_t u \, dt = \int_\Omega \partial_t^2 u \cdot \partial_t u \, dt = (\partial_t^2 u, \partial_t u).$$

Thus the time variation is zero and therefore by integration in time yields

$$\frac{1}{2}\int_0^t \frac{d}{dt}\|\partial_t u\|^2 + \frac{1}{2}\int_0^t \frac{d}{dt}\|\nabla u\|^2 = 0.$$

Thus,

$$\underbrace{\underbrace{\|\partial_t u(t)\|^2}_{=E_{kin}(t)} + \underbrace{\|\nabla u(t)\|^2}_{=E_{pot}(t)}}_{=E_{tot}(t)} = const = \underbrace{\underbrace{\|v_0\|^2}_{=E_{kin}(0)} + \underbrace{\|\nabla u_0\|^2}_{=E_{pot}(0)}}_{=E_{tot}(0)}$$

which is energy conservation

$$E_{tot}(t) = E_{tot}(0)$$

at all times $t \in I$.

Consequently, the construction of **good** time-stepping schemes becomes more involved than for the parabolic case. From the ODE analysis, we recall that for the backward Euler scheme and the Crank-Nicolson scheme are both implicit and unconditionally stable. But the backward Euler scheme is very dissipative and may introduce too much artificial diffusion alternating the energy conservation on the discrete time level. Here, the Crank-Nicolson scheme is much better suited. We recall in other words from Section 12.3.8:

Definition 12.67 (Stability and properties of time stepping schemes)**.** *A good time stepping scheme for time-discretizing partial differential equations should satisfy:*

- *A-stability (local convergence): an example is the Crank-Nicolson scheme (trapezoidal rule);*
- *strict A-stability (global convergence): an example is the shifted Crank-Nicolson scheme;*
- *strong A-stability (smoothing property): examples are backward Euler (in particular even L-stable[10]) and Fractional-step-θ schemes;*
- *small dissipation (energy conservation).*

◇

Let us now describe the properties of the previous schemes in more detail:

- The explicit Euler scheme is cheap in the computational cost since no equation system needs to be solved. However, in order to be numerically stable (very, very) small time steps must be employed, which makes this scheme infeasible for most PDE problems
- A classical scheme for problems with a stationary limit is the (implicit) backward Euler scheme (BE), which is strongly A-stable (but only from first order), robust and dissipative. It is used in numerical Examples, where a stationary limit must be achieved. However, due to its dissipativity this schemes dampes high-frequent temporal parts of the solution too much and for this reason this scheme is not recommended for nonstationary PDE problems.

[10] A time-stepping scheme is L-stable if it is A-stable and the stability function satisfies $|R(z)| \to 0$ for $z \to \infty$. In this respect the Crank-Nicolson scheme is not L-stable since $R(z) \to 1$ for $z \to -\infty$.

- In contrast, the (implicit) Crank-Nicolson scheme is of second order, A-stable, and has very little dissipation but suffers from case-to-case instabilities caused by rough initial and/or boundary data. These properties are due to weak stability (it is not *strongly* A-stable). A variant of the Crank-Nicolson scheme is called *shifted* Crank-Nicolson scheme, is analyzed in Rannacher et al. [74, 100], which allows for global stability of the solution [11]. In particular, Rannacher analyzed in [99] the shifted Crank-Nicolson scheme for linear evolution equations and how they can be modified in order to make them suitable for long-term computations (without reducing second order accuracy!!).
- The fourth scheme summarizes the advantages of the previous two and is known as the Fractional-Step-θ scheme for computing unsteady-state simulations [58]. Roughly-speaking it consists of summarizing three Crank-Nicolson steps and has therefore the same accuracy and compuational cost as the Crank-Nicolson scheme. However, it is more robust, i.e., it is strongly A-stable (as backward Euler) but has 2nd order accuracy as Crank-Nicolson, and is therefore well-suited for computing solutions with rough initial data and long-term computations for problems. This property also holds for ALE-transformed fluid equations, which is demonstrated in a numerical test below. We also refer the reader to a modification of the Fractional-Step-θ scheme [133].

12.6.5 One-Step-θ times-stepping for the wave equation

To finish, we aim to apply a One-Step-θ scheme and therefore, we need to work with the first-order mixed-system. Then, we apply the One-Step-θ scheme to Formulation 12.60 idea to discretize in time: Find v^n and u^n such that:

$$(\rho\frac{v^n - v^{n-1}}{k}, \phi) + \theta(\nabla u^n, \nabla\phi) + (1-\theta)(\nabla u^{n-1}, \nabla\phi) = \theta(f^n, \phi) + (1-\theta)(f^{n-1}, \phi)$$
$$(\frac{u^n - u^{n-1}}{k}, \psi) = \theta(v^n, \psi) + (1-\theta)(v^{n-1}, \psi)$$

And finally, we discretize in space using a finite element scheme: Find v_h^n and u_h^n such that:

$$(\rho\frac{v_h^n - v_h^{n-1}}{k}, \phi_h) + \theta(\nabla u_h^n, \nabla\phi_h) + (1-\theta)(\nabla u_h^{n-1}, \nabla\phi_h) = \theta(f^n, \phi_h) + (1-\theta)(f^{n-1}, \phi_h)$$
$$(\frac{u_h^n - u_h^{n-1}}{k}, \psi_h) = \theta(v_h^n, \psi_h) + (1-\theta)(v_h^{n-1}, \psi_h)$$

Re-arringing given data terms on the right hand side yields:

$$k^{-1}(\rho v_h^n, \phi_h) + \theta(\nabla u_h^n, \nabla\phi_h) = k^{-1}(\rho v_h^{n-1}, \phi_h) + \theta(f^n, \phi_h) + (1-\theta)(f^{n-1}, \phi_h) - (1-\theta)(\nabla u_h^{n-1}, \nabla\phi_h)$$
$$k^{-1}(u_h^n, \psi_h) - \theta(v_h^n, \psi_h) = k^{-1}(u_h^{n-1}, \psi_h) + (1-\theta)(v_h^{n-1}, \psi_h)$$

To obtain the system matrix and the linear system, we proceed as in the parabolic case by considering that we now seek two solutions u_h^n and v_h^n.

With the previous discretizations, we can now define

Algorithm 12.68. *Given the initial guesses u_h^0 and v_h^0 and given at each time the right hand side forces $f^n := f(t^n)$ and $k_n = t^n - t^{n-1}$ we solve for each $n = 1, 2, 3, \ldots, N$: Find v_h^n and u_h^n such that*

$$k_n^{-1}(\rho v_h^n, \phi_h) + \theta(\nabla u_h^n, \nabla\phi_h) = k_n^{-1}(\rho v_h^{n-1}, \phi_h) + \theta(f^n, \phi_h) + (1-\theta)(f^{n-1}, \phi_h) - (1-\theta)(\nabla u_h^{n-1}, \nabla\phi_h)$$
$$k_n^{-1}(u_h^n, \psi_h) - \theta(v_h^n, \psi_h) = k_n^{-1}(u_h^{n-1}, \psi_h) + (1-\theta)(v_h^{n-1}, \psi_h)$$

for $(\phi_h, \psi_h) \in V_h \times W_h$ with $V_h \times W_h \subset H_0^1 \times L^2$.

Remark 12.69. *An abstract generalization of the One-Step-θ scheme has been formulated in [135][Chapter 5].*

[11] A famous alternative to the shifted Crank-Nicolson scheme is Rannacher's time-marching technique in which the unshifted Crank-Nicolson scheme is augmented with backward Euler steps for stabilization and in particular for irregular initial data [99]. Adding backward Euler steps within Crank-Nicolson does also stabilize during long-term computations [74, 99].

We now use the FEM representations at time t^n

$$u_h^n = \sum_{j=1}^{N_1} a_j \varphi_j$$
$$v_h^n = \sum_{j=N_1+1}^{N_2} b_j \psi_j$$

and obtain:

$$\sum_j b_j k_n^{-1}(\rho\psi_j, \varphi_i) + \sum_j a_j \theta(\nabla\varphi_j, \nabla\varphi_i) = k_n^{-1}(\rho v_h^{n-1}, \varphi_i) + \theta(f^n, \varphi_i) + (1-\theta)(f^{n-1}, \varphi_i) - (1-\theta)(\nabla u_h^{n-1}, \nabla\varphi_i)$$
$$\sum_j a_j k_n^{-1}(\varphi_j, \psi_i) - \sum_j b_j \theta(\psi_j, \psi_i) = k_n^{-1}(u_h^{n-1}, \psi_i) + (1-\theta)(v_h^{n-1}, \psi_i)$$

resulting in the following block system (respectively k, θ, and ρ):

$$\begin{pmatrix} A_{uu} & B_{uv} \\ B_{vu} & M_{vv} \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} F \\ G \end{pmatrix}$$

with

$$A_{uu} = (\nabla\varphi_j, \nabla\varphi_i)_{ij}, \quad B_{uv} = (\psi_j, \varphi_i)_{ij}, \quad B_{vu} = (\varphi_j, \psi_i)_{ij}, \quad M_{vv} = (\psi_j, \psi_i)_{ij}$$

and

$$a = (a_j)_{j=1}^{N_1}, \quad b = (b_j)_{j=1}^{N_2}.$$

12.6.6 Stability analysis / energy conservation on the time-discrete level

Similar to parabolic problems, we perform a stability analysis, but this time with a focus on energy conservation on the time-discretized level.

We set

$$0 = t_0 < t_1 < \ldots < t_N = T, \quad k = t_n - t_{n-1}$$

The model problem is:

Formulation 12.70. *Let $V = H_0^1$ and $W = L^2$. Given $u^{n-1} \in V$ and $v^{n-1} \in W$ we solve for $n = 1, \ldots, N$ using the Crank-Nicolson scheme: Find $(u^n, v^n) \in V \times W$ such that*

$$(v^n - v^{n-1}, \phi) + \frac{1}{2}k(\nabla u^n + \nabla u^{n-1}, \nabla\phi) = \frac{1}{2}(f^n + f^{n-1}, \phi) \quad \forall \phi \in V,$$
$$(u^n - u^{n-1}, \psi) - \frac{1}{2}k(v^n + v^{n-1}, \psi) = 0 \quad \forall \psi \in W.$$

Proposition 12.71. *Setting $f^n = 0$ for all $n = 1, \ldots, N$ in the previous formulation, we have*

$$\underbrace{\|\nabla u^n\|^2}_{E_{pot}^n} + \underbrace{\|v^n\|^2}_{E_{kin}^n} = \underbrace{\|\nabla u^{n-1}\|^2}_{E_{pot}^{n-1}} + \underbrace{\|v^{n-1}\|^2}_{E_{kin}^{n-1}}$$

and therefore

$$E_{tot}^n = E_{tot}^{n-1}.$$

Proof. We start from Formulation 12.70 and choose once again clever test functions:

$$\phi = u^n - u^{n-1}, \quad \psi = v^n - v^{n-1}.$$

Then:

$$(v^n - v^{n-1}, u^n - u^{n-1}) + \frac{1}{2}k(\nabla u^n + \nabla u^{n-1}, \nabla(u^n - u^{n-1})) = 0,$$
$$(u^n - u^{n-1}, v^n - v^{n-1}) - \frac{1}{2}k(v^n + v^{n-1}, v^n - v^{n-1}) = 0.$$

We subtract the second equation from the first equation. Due to the bi-linear property the first terms cancel each other:

$$\frac{1}{2}k(\nabla u^n + \nabla u^{n-1}, \nabla(u^n - u^{n-1})) + \frac{1}{2}k(v^n + v^{n-1}, v^n - v^{n-1}) = 0.$$

Next, the cross terms cancel due to different signs and it remains:

$$\frac{1}{2}k(\nabla u^n, \nabla u^n) + \frac{1}{2}k(v^n, v^n) = \frac{1}{2}k(\nabla u^{n-1}, \nabla u^{n-1}) + \frac{1}{2}k(v^{n-1}, v^{n-1}),$$

therefore we easily see

$$\|\nabla u^n\| + \|v^n\| = \|\nabla u^{n-1}\| + \|v^{n-1}\|.$$

Voilà! We are finished. □

We next show that indeed the implicit Euler scheme introduces artificial viscosity and loose energy over time:

Formulation 12.72. *Let $V = H_0^1$ and $W = L^2$. Given $u^{n-1} \in V$ and $v^{n-1} \in W$ we solve for $n = 1, \dots, N$ using the implicit Euler scheme: Find $(v^n, u^n) \in W \times V$ such that*

$$\begin{aligned}(v^n - v^{n-1}, \phi) + k(\nabla u^n, \nabla\phi) &= (f^n, \phi) \quad \forall \phi \in V,\\ (u^n - u^{n-1}, \psi) - k(v^n, \psi) &= 0 \quad \forall \psi \in W.\end{aligned}$$

Proposition 12.73. *Setting $f^n = 0$ for all $n = 1, \dots, N$ in the previous formulation, we have*

$$\underbrace{\|\nabla u^n\|^2}_{E^n_{pot}} + \underbrace{\|v^n\|^2}_{E^n_{kin}} \leq \underbrace{\|\nabla u^{n-1}\|^2}_{E^{n-1}_{pot}} + \underbrace{\|v^{n-1}\|^2}_{E^{n-1}_{kin}}$$

and therefore

$$E^n_{tot} \leq E^{n-1}_{tot}.$$

Therefore, energy is dissipated due to the 'wrong' numerical scheme.

Proof. The proof is basically the same as before. We start from Formulation 12.72 and choose the same test functions as before:

$$\phi = u^n - u^{n-1}, \quad \psi = v^n - v^{n-1}.$$

Then:

$$\begin{aligned}(v^n - v^{n-1}, u^n - u^{n-1}) + k(\nabla u^n, \nabla(u^n - u^{n-1})) &= 0,\\ (u^n - u^{n-1}, v^n - v^{n-1}) - k(v^n, v^n - v^{n-1}) &= 0.\end{aligned}$$

We subtract the second equation from the first equation. Due to the bi-linear property the first terms cancel each other:

$$k(\nabla u^n, \nabla(u^n - u^{n-1})) + k(v^n, v^n - v^{n-1}) = 0.$$

Next,

$$k(\nabla u^n, \nabla u^n) - k(\nabla u^n, \nabla u^{n-1}) + k(v^n, v^n) - k(v^n, v^{n-1}) = 0.$$

Here, bring the cross terms on the right hand side and apply once again Youngs' inequality:

$$k(\nabla u^n, \nabla u^n) + k(v^n, v^n) \leq \frac{1}{2}\|\nabla u^n\| + \frac{1}{2}\|\nabla u^{n-1}\| + \frac{1}{2}\|v^n\| + \frac{1}{2}\|\nabla v^{n-1}\|.$$

Re-arranging terms yields the assertion. □

12.6.7 Convergence results

A priori and a posteriori error estimates for the linear second-order in time wave equation using the backward Euler scheme can be found [17]. Specifically, adaptive time step sizes can be chosen therein. Other results for uniform time steps, but including the Crank-Nicolson scheme are proven in [64].

12.6.8 Summary of stability, convergence and energy conservation

We summarize our findings (see for more details in [8, 17, 64]). Our first conclusions is: In extension to elliptic and parabolic problems, the numerical discretization of the wave equation asks for temporal schemes that satisfy energy conservation.

Summarizing everything yields:

- *Stability in the L^2-norm:* the One-Step-θ scheme is unconditionally stable, i.e., there is no time step restriction on k if and only if $\theta \in [\frac{1}{2}, 1]$.
- *Convergence* It holds (can be proven using tools from ODE numerical analysis):
  - $\theta \in [0, 0.5) \cup (0, 5, 1]$: linear convergence $O(k)$
  - $\theta = 0.5$: quadratic convergence $O(k^2)$.
- *Energy conservation:* the one-step-θ scheme preserves energy only for the choice $\theta = \frac{1}{2}$. For $\theta > \frac{1}{2}$ (e.g., the implicit Euler scheme for the choice $\theta = 1$) the scheme dissipates energy as shown in the previous subsection. For $\theta < \frac{1}{2}$ energy conservation depends, but in all cases schemes will be unstable.

Consequently, the Crank-Nicolson scheme is an optimal time-stepping scheme for hyperbolic equations.

When explicit schemes are taken, possible restrictions with respect to the time-step size are weaker for hyperbolic problems than for parabolic differential equations [64]; namely

Definition 12.74 (Courant-Friedrichs-Levy - CFL)**.** *A necessary condition for explicit time stepping schemes is the Courant-Friedrichs-Levy (CFL) condition, i.e., the time step size k is dependent on material parameters and the spatial discretization parameter h:*

- *Parabolic problems: $k \leq \frac{1}{2}ch^2$;*
- *Hyperbolic problems: $k \leq a^{-1}h$, where a is of order of the elasticity constant in the wave operator.*

12.7 Numerical tests: scalar wave equation

We provide some impressions about the scalar-valued wave equation. The setting is similar to Section 8.16.1. We take $T = 10s$ and $k = 1s$ using the Crank-Nicolson scheme. The initial solutions u^0 and v^0 are taken to be zero. In the Figures 55 and 56 we show the evolution of the displacements and the velocity.

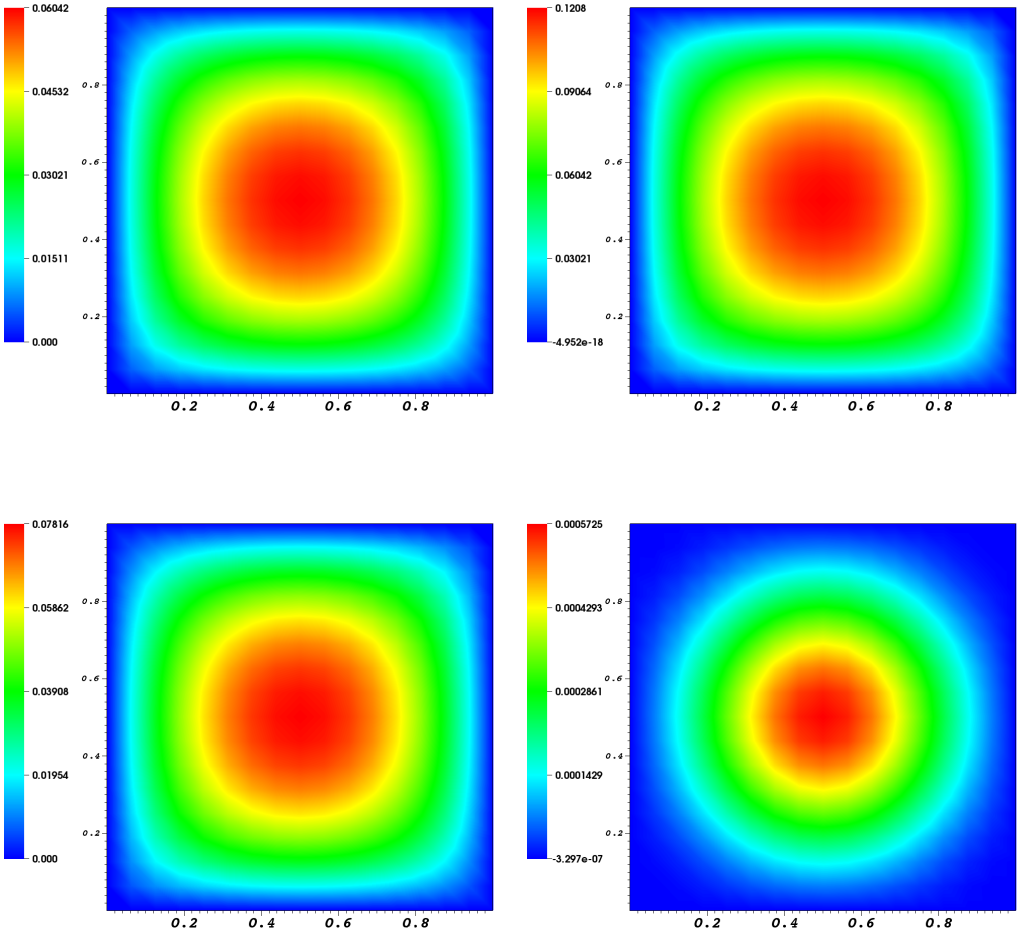


Figure 55: Scalar wave equation: Evolution of displacements (left) and velocities (right).

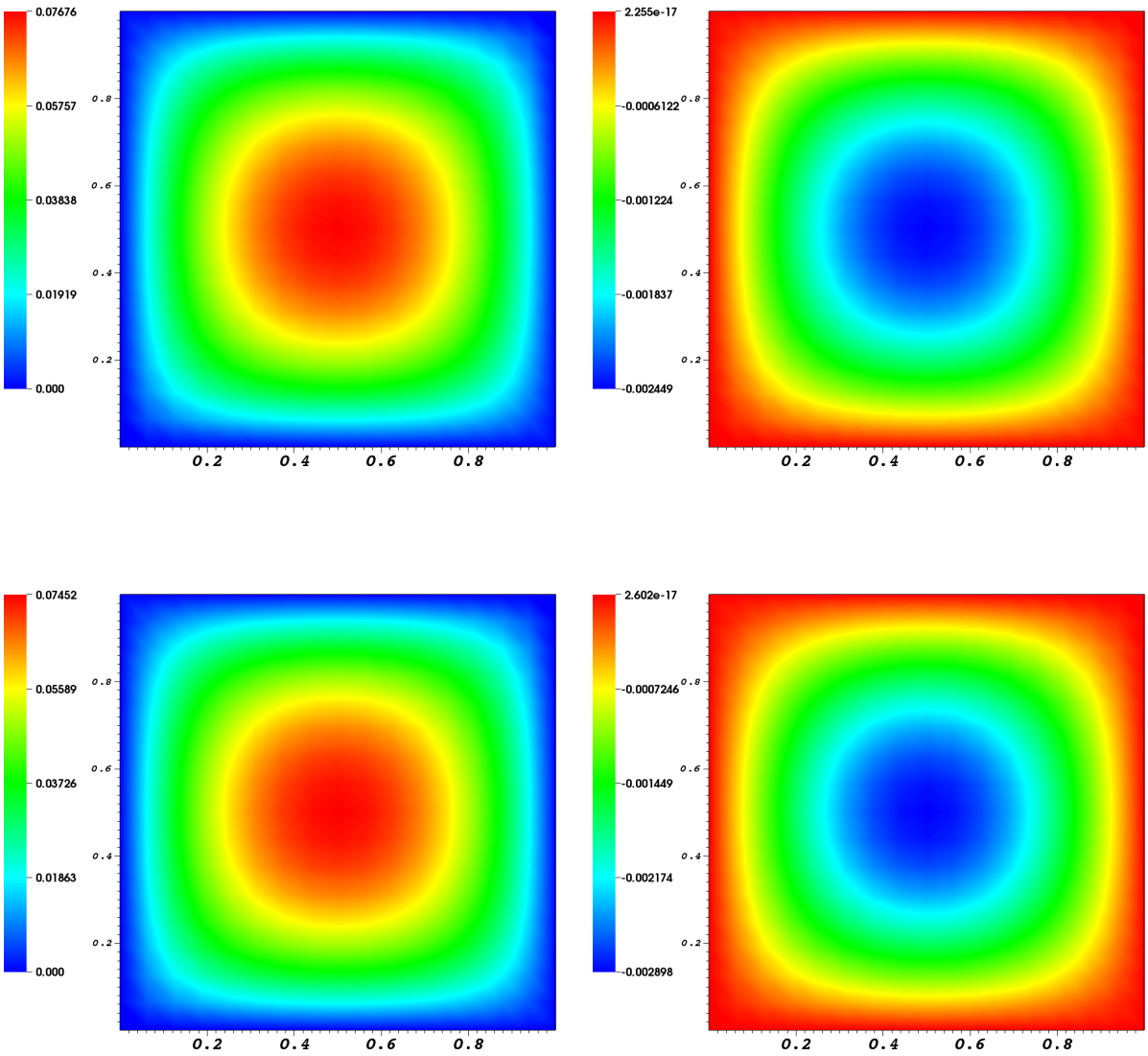


Figure 56: Scalar wave equation: Evolution of displacements (left) and velocities (right).

12.8 Numerical tests: elastic wave equation

We extend the scalar wave equation to the elastic wave equation. Specifically, the linearized elastic wave equation is augmented with the acceleration term:

Problem 12.75 (Discretized nonstationary linearized elasticity / Elastodynamics)**.** *Given the initial conditions u_0 and v_0 and a vector-valued right hand side $\hat{f} = (f_x, f_y)$. Find $(\hat{u}_h, \hat{v}_h) \in \hat{V}_h^0 \times \hat{W}_h$ such that*

$$(\hat{\rho}\partial_t \hat{v}_h, \hat{\varphi}_h) + (\widehat{\Sigma}_{h,lin}, \widehat{\nabla}\hat{\varphi}_h) = (\hat{f}, \hat{\varphi}_h) \quad \forall \hat{\varphi}_h \in \hat{V}_h^0,$$
$$(\partial_t \hat{u}_h, \hat{\psi}_h) - (\hat{v}_h, \hat{\psi}_h) = 0 \quad \forall \hat{\psi}_h \in \hat{W}_h,$$

where $\hat{V}_h^0 \subset V := \{u \in H^1 |\ u = 0 \text{ on } \Gamma_{left}\}$, here Γ_{left} is the left boundary of the specimen and and $\hat{W}_h \subset L^2$ and

$$\widehat{\Sigma}_{h,lin} = 2\mu \widehat{E}_{h,lin} + \lambda tr\widehat{E}_{h,lin}\hat{I},$$

and the identity matrix $\hat{I}$ and

$$\widehat{E}_{h,lin} = \frac{1}{2}(\widehat{\nabla}\hat{u}_h + \widehat{\nabla}\hat{u}_h^T).$$

12.8.1 Constant force - stationary limit

First, we redo the 2D test from Section 11.4 in which the left boundary is fixed and $\hat{\rho} = 1$ and $(f_x, f_y) = (0, -9.81)$. These findings are displayed in Figure 57. As time-stepping scheme, the Crank-Nicolson scheme, i.e., $\theta = 0.5$ is used. The time step size is $k = 1s$ and the total time is $T = 10s$. The initial solutions are $u_0 = 0$ and $v_0 = 0$.

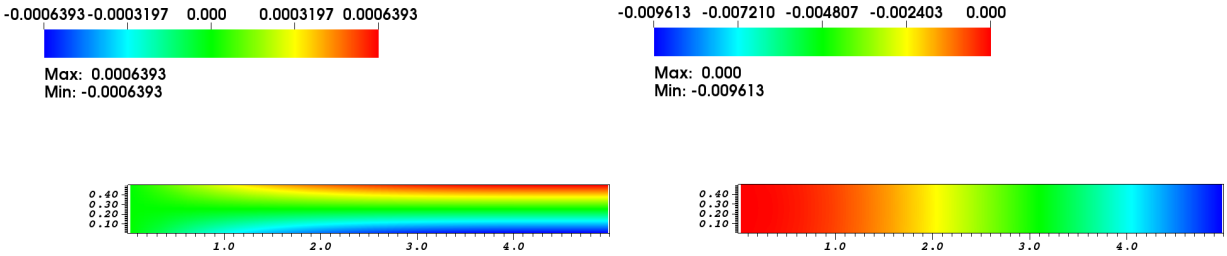


Figure 57: 2D nonstationary linearized elasticity with a constant right hand side $(f_x, f_y) = (0, -9.81)$. At left, the u_x solution is displayed. At right, the u_y solution is displayed.

12.8.2 Time-dependent force - Crank-Nicolson scheme

In the second test, we prescribe the right hand is as

$$(f_x^n, f_y^n) = (0, \sin(t))^T,$$

with the uniform time step size $k = 0.02s$ and $N = 314$ (that is 314 time step solutions). The total time is $T = k * N = 6.28s$. The results at $N = 2,157,314$ are displayed in Figure 58.

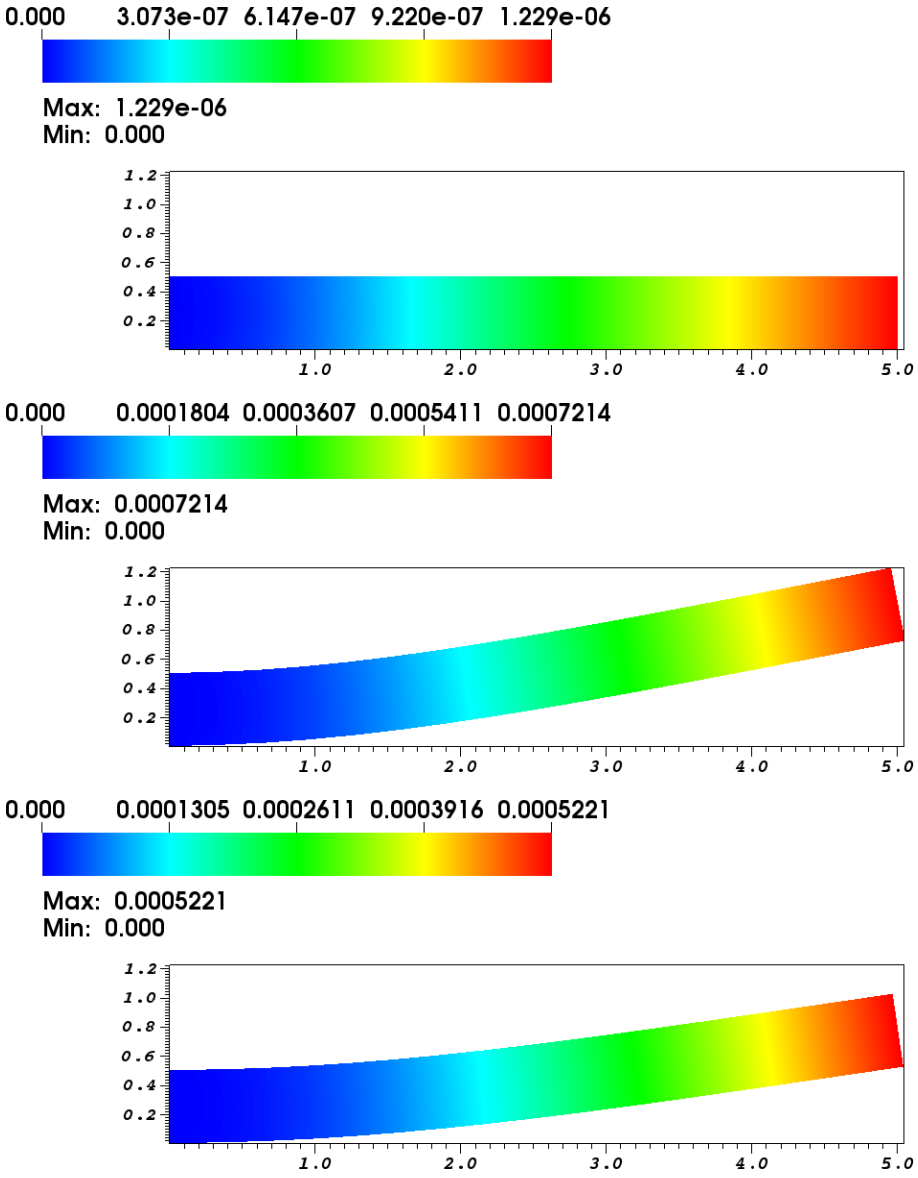


Figure 58: 2D nonstationary linearized elasticity with $(f_x^n, f_y^n) = (0, \sin(t))^T$. The u_y solution is displayed at $N = 2,157,314$. The displacements are amplified by a factor of 1000 such that a visible deformation of the elastic beam can be seen. Here, the Crank-Nicolson scheme was used.

12.8.3 Time-dependent force - backward Euler scheme

In the third test, we use the backward Euler scheme. From the theory we know that this scheme is too dissipative and introduces artificial numerical damping. Indeed, observing Figure 59, we see that the displacements at $N = 157$ and $N = 314$ are much smaller than in the corresponding Crank-Nicolson test case.

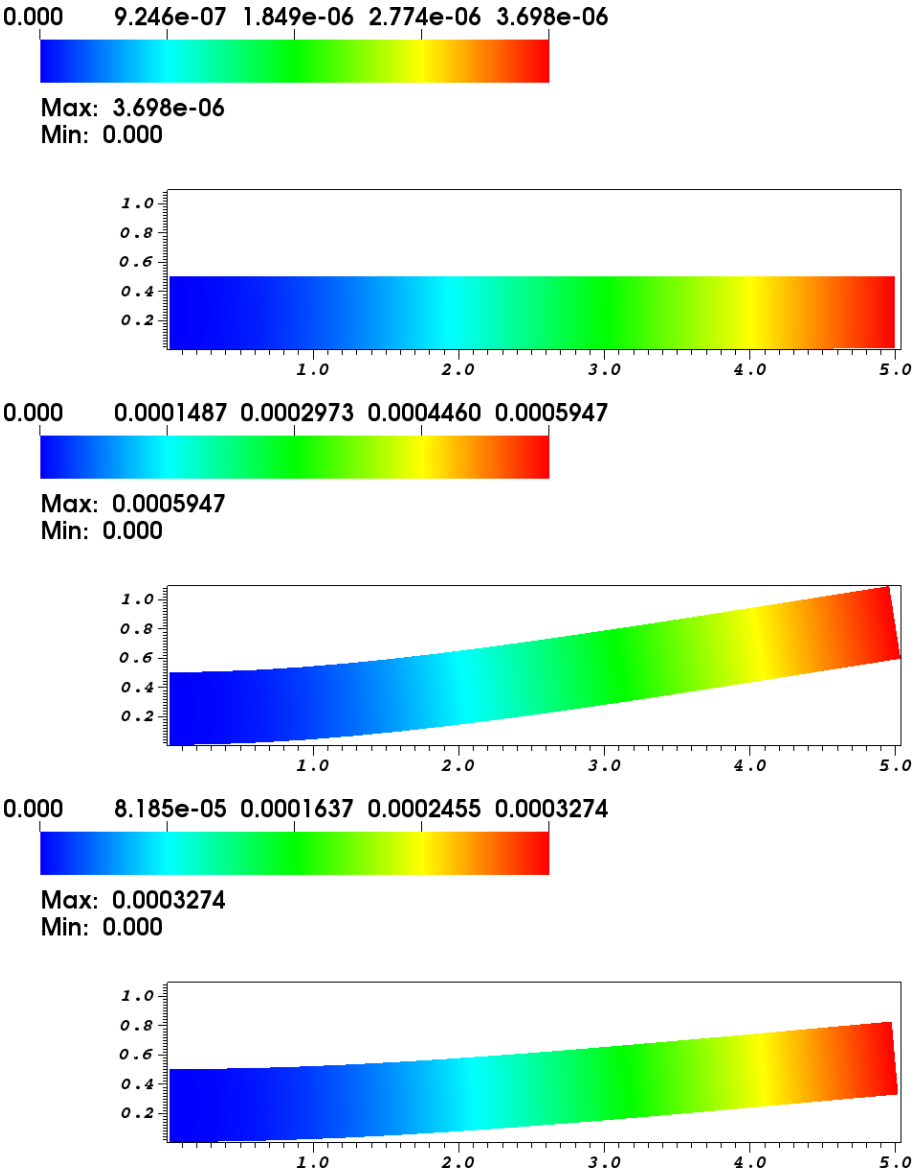


Figure 59: 2D nonstationary linearized elasticity with $(f_x^n, f_y^n) = (0, \sin(t))^T$. The u_y solution is displayed at $N = 2, 157, 314$. The displacements are amplified by a factor of 1000 such that a visible deformation of the elastic beam can be seen. Here, the backward Euler scheme was used.

12.9 Numerical tests: elastic wave equation in a flow field

We go to [140]. The problem is fluid-structure interaction (Section 14.9), but this is not important at all here. What is important is the fact that the elastic part is described by a (nonlinear) wave equation (again Section 14.9). The configuration is a cylinder that is shifted such that the elastic beam (should!) start oscillating.

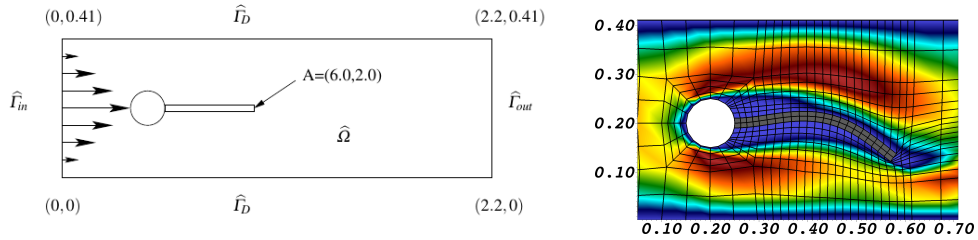


Figure 60: Left: Flow around a cylinder with an elastic beam described by a nonlinear wave equation [77]. Right: Snapshot of the numerical solution of the velocity field. Both figures copied from [140].

We learned previously that the Crank-Nicolson scheme should be an ideal candidate for problems with wave phenomena. First, we want to refer to 2.7 (taken from the author's PhD thesis [135]) that the pure Crank-Nicolson scheme with $\theta = 0.5$ will be subject to stability issues after some time. Let us now work with the shifted variant $\theta = 0.5 + k$ and investigate different k towards the implicit side. The largest choice will be $k = 0.1$, i.e., $\theta = 0.5 + 0.1 = 0.6$. The results are displayed in Figure 61. We nicely see that from $k = 0.05$ (i.e., $\theta = 0.55$) the numerical dissipation is already too high and damps significantly the (physical!) oscillations of the elastic beam. For $\theta = 0.6$, the beam will even not start to move! Thus, $\theta \geq 0.6$, specifically $\theta = 1$ (backward Euler), are useless schemes for these situations because the numerical dissipation is too high and energy on the time-discrete level is not conserved.

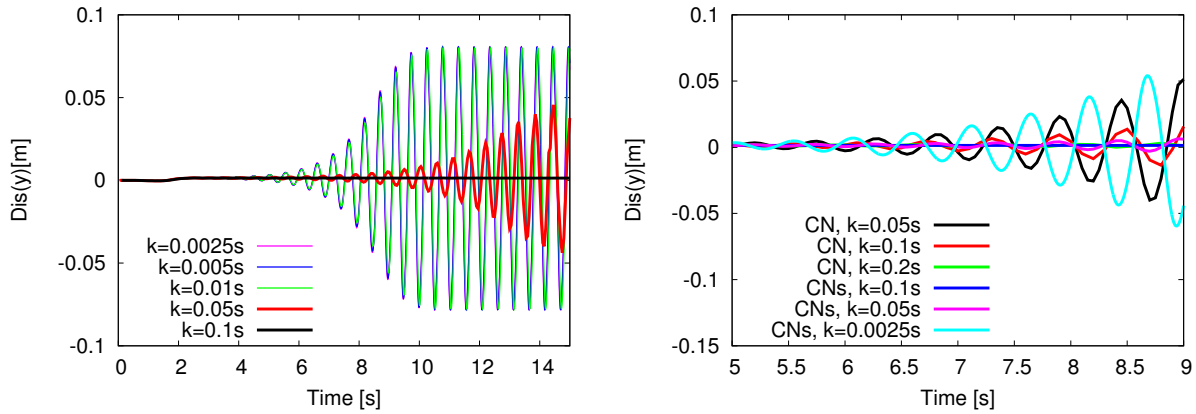


Figure 61: Taken from [140]: Comparison of u_y (tip of the elastic beam in y direction) on mesh level 1 and different time step sizes k using the shifted Crank-Nicolson (CNs) scheme (left) and a comparison with the classical Crank-Nicolson (CN) method (right). First thanks to the monolithic formulation and implicit time-stepping we are able to use large time steps, e.g., $k = 0.2s$. Secondly, the large time step is not sufficient any more to lead to the correct oscillations of the elastic beam. In particular, care of the correct choice of the damping factor using the shifted version must be taken. From this figure, we infer that the largest time step size is around $k \sim 0.01s$ in order to obtain the correct amplitude of oscillations.

12.10 Chapter summary and outlook

In this chapter, we have investigated time-dependent problems. In the next chapter we leave time-dependence and go back to stationary problems, but include now nonlinear phenomena. These lead to fascinating research topics for which theory and discretizations only have partially developed, and therefore constitute also current research directions.

13 Nonlinear problems

Most problems are nonlinear and such settings are really fascinating to study. For this reason, a first class on numerical methods for partial differential equations, with the risk of being dense, should at least strive how nonlinear problems can be approached. This chapter and Chapter 14 have been partially re-written and are both complementary to my book on multiphysics phase-field fracture [141] in which nonlinear problems and coupled variational inequality systems (CVIS) are studied in great detail.

13.1 Nonlinear PDEs: strong forms

We introduce nonlinear problems in terms of the so-called p-Laplace problem, which has important applications, but reduces to the Poisson problem for $p = 2$. This setting has no time dependence.

13.1.1 p-Laplace

Find $u : \Omega \to \mathbb{R}$:

$$-\nabla \cdot (|\nabla u|^{p-2}\nabla u)^{p/2} = f \tag{267}$$

Properties: nonlinear (quasilinear), stationary, scalar-valued. We need $W^{k,p}$ Sobolev spaces (Section 7.1.7).

13.1.2 Semilinear equation

Find $u : \Omega \to \mathbb{R}$:

$$-\Delta u + u^2 = f \tag{268}$$

Properties: nonlinear (semilinear), stationary, scalar-valued.

13.1.3 Incompressible Euler equations

Find $v : \Omega \to \mathbb{R}^n$ and $p : \Omega \to \mathbb{R}$

$$\partial_t + (v \cdot \nabla)v + \nabla p = f, \quad \nabla \cdot v = 0 \tag{269}$$

Properties: nonlinear (quasilinear), nonstationary, vector-valued, PDE system.

13.1.4 Incompressible Navier-Stokes equations

Find $v : \Omega \to \mathbb{R}^n$ and $p : \Omega \to \mathbb{R}$

$$\partial_t v + (v \cdot \nabla)v - \frac{1}{Re}\Delta v + \nabla p = f, \quad \nabla \cdot v = 0 \tag{270}$$

with Re being the Reynolds' number. Properties: semilinear, ... For $Re \to \infty$ we obtain the Euler equations. Properties: nonlinear (quasilinear), nonstationary, vector-valued, PDE system.

13.1.5 A coupled nonlinear problem (I)

Find $u : \Omega \to \mathbb{R}$ and $\varphi : \Omega \to \mathbb{R}$

$$-\nabla \cdot (a(\varphi)\nabla u) = f, \tag{271}$$

$$a(\varphi)|\nabla u|^2 - \Delta\varphi = g \tag{272}$$

Properties: nonlinear, coupled problem via coefficients, stationary. Equations become linear when solution variables are fixed in the other equation.

13.1.6 A coupled nonlinear problem (II)

Find $u : \Omega \to \mathbb{R}$ and $\varphi : \Omega \to \mathbb{R}$

$$-\Delta u = f(\varphi), \tag{273}$$
$$|\nabla u|^2 - \Delta \varphi = g(u) \tag{274}$$

Properties: nonlinear, coupled problem via right hand sides, stationary. Equations become linear when solution variables are fixed in the other equation.

13.1.7 A coupled nonlinear problem (III)

Let Ω_1 and Ω_2 with $\Omega_1 \cap \Omega_2 = 0$ and $\bar{\Omega}_1 \cap \bar{\Omega}_2 = \Gamma$ and $\bar{\Omega}_1 \cup \bar{\Omega}_2 = \Omega$. Find $u_1 : \Omega_1 \to \mathbb{R}$ and $u_2 : \Omega_2 \to \mathbb{R}$:

$$-\Delta u_1 = f_1 \quad \text{in } \Omega_1, \tag{275}$$
$$-\Delta u_2 = f_2 \quad \text{in } \Omega_2, \tag{276}$$
$$u_1 = u_2 \quad \text{on } \Gamma, \tag{277}$$
$$\partial_n u_1 = \partial_n u_2 \quad \text{on } \Gamma, \tag{278}$$
$$\tag{279}$$

Properties: linear, coupled problem via interface conditions, stationary.

13.2 Nonlinear PDEs: weak forms

The notation for weak forms is via semi-linear forms:

$$a(u)(\phi)$$

where the (solution) variable u is nonlinear and the test function ϕ is linear. The nonlinear problem statement in abstract form reads

Formulation 13.1. *Find $u \in V$ such that*

$$a(u)(\phi) = l(\phi) \quad \forall \phi \in V,$$

with the right hand side linear form $l(\phi) = (f, \phi)$ with a given force f.

Due to the nonlinearity in u, simply inserting

$$u_h = \sum_{j=1}^{n} u_j \phi_j$$

yields a nonlinear equation system. Consider as example

$$(\nabla u, \nabla \phi) + (u^2, \phi) = (f, \phi)$$

Then, the discrete form reads

$$(\nabla u_h, \nabla \phi_h) + (u_h^2, \phi_h) = (f, \phi_h)$$

and inserting the linear combination yields

$$\sum_{j=1}^{n} u_j (\nabla \phi_j, \nabla \phi_i) + \left(\left(\sum_{j=1}^{n} u_j \phi_j \right)^2, \phi_i \right) = (f, \phi_i)$$

for $i = 1, \ldots, n$. For this reason, we need to linearize as we show in Section 13.7.

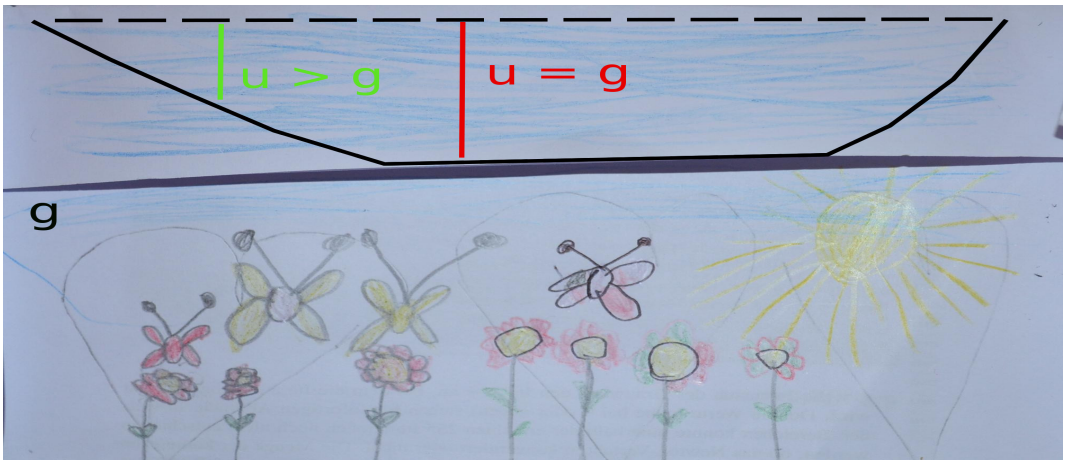


Figure 62: Obstacle problem $-u'' = -1$ in Ω and $u(0) = u(1) = 0$: deformation u of a line that is constraint by the obstacle g.

13.3 A variational inequality I: Obstacle problem

The obstacle problem is the prototype of a variational inequality and is very classical and intensively investigated in [83] and [84]. We consider an elastic membrane in $\mathbb{R}^n$ with $n = 2$ or simpler $n = 1$ resulting in the clothesline problem. We want avoid that the wet clothes touch the flowers as illustrated in Figure 62.

Let $\Omega \subset \mathbb{R}$ (all developements in this section hold also for $\Omega \subset \mathbb{R}^n$) be open and $u : \Omega \to \mathbb{R}$ and $f : \Omega \to \mathbb{R}$. The (nonlinear) potential energy is defined as

$$E(u) = \int_\Omega \left(\mu(\sqrt{1 + |\nabla u|^2} - 1) - fu\right) dx$$

where $\mu > 0$ is a material parameter. Taylor linearization yields:

$$E(u) = \frac{1}{2} \int_\Omega (\mu |\nabla u|^2 - fu)\, dx$$

The physical state without constraints is given by

$$\min_{u \in V} E(u)$$

with $V = \{v \in H^1(\Omega) | v = u_0 \text{ on } \partial\Omega\}$.

We introduce the constraint (for instance a table that blocks the further deformation of the membrane):

$$u \geq g, \quad g \in L^2$$

Then:

$$\min_{u \in V, u \geq g} E(u)$$

The admissible space is the convex set

$$K = \{v \in H^1 | v = u_0 \text{ on } \partial\Omega,\ v \geq g \text{ in } \Omega\}.$$

We recall the definition of a convex set:

$$u, v \in K : \quad \theta u + (1 - \theta)v \in K.$$

Formulation 13.2. *If $u \in K$ is a minimum, it holds:*

$$E(u) = \min_{v \in K} E(v).$$

13.3.0.1 Variational formulation We now derive a variational formulation. Let $\theta v+(1-\theta)u = u+\theta(v-u) \in K$ for $\theta \in [0,1]$. Then it clearly holds:

$$E(u+\theta(v-u)) \geq E(u)$$

We derive now the first-order optimality condition (i.e., the PDE in weak form):

- Differentiate w.r.t. θ;
- Set $\theta = 0$.

Remark 13.3. *For the differentiation in Banach spaces or $\mathbb{R}^n$ we refer the reader to Section 7.4.*

Then:

$$\begin{aligned}
\frac{d}{d\theta}E(u+\theta(v-u))|_{\theta=0} &\geq \frac{d}{d\theta}E(u) \\
\Leftrightarrow \frac{d}{d\theta}E(u+\theta(v-u))|_{\theta=0} &\geq 0 \\
\Leftrightarrow \frac{d}{d\theta}\int_\Omega \mu\nabla u\cdot\nabla(u+\theta(v-u))|_{\theta=0}\,dx - \int_\Omega f(v-u)\,dx &\geq 0 \\
\Rightarrow \int_\Omega \mu\nabla u\cdot\nabla(v-u)\,dx - \int_\Omega f(v-u)\,dx &\geq 0
\end{aligned}$$

for all $v \in K$.

In summary:

Formulation 13.4 (Obstacle problem: variational formulation)**.** *We have*

$$(\mu\nabla u, \nabla(v-u)) \geq (f, v-u)$$

for $K = \{v \in H^1 | v = u_0 \text{ on } \partial\Omega,\ v \geq g \text{ in } \Omega\}$.

13.3.0.2 Strong formulation Using the fundamental lemma of calculus of variations, we derive as usually the strong form:

$$\begin{aligned}
-\nabla\cdot(\mu\nabla u) &\geq f \quad \text{in } \Omega, \\
u &\geq g \quad \text{in } \Omega, \\
(\nabla\cdot(\mu\nabla u)+f)(u-g) &= 0 \quad \text{in } \Omega, \\
u &= u_0 \quad \text{on } \partial\Omega \\
u &= g \quad \text{on } \Gamma \\
\mu\nabla u\cdot n &= \mu\nabla g\cdot n \quad \text{on } \Gamma.
\end{aligned}$$

The boundary Γ is a so-called **free boundary**. The third condition is the so-called complementarity condition that links the PDE and the obstacle condition.

We define the subregions as

- $A = \Omega \setminus N$: active set or contact zone (we sit on the obstacle), defined by

$$A = \{x \in \Omega :\ u = g\}$$

- N is the inactive set (we solve the PDE)
- $\Gamma = \partial A \cap \partial N$ is the free boundary.

Remark 13.5. *We notice that the two inner conditions of the free boundary are exactly of the same nature as the interface conditions of coupled problems in different subdomains; see Section 14.1. Therefore, we have a natural link between the obstacle problem and surface-coupled PDEs.*

13.3.0.3 Lagrange multiplier formulation We can also explicitly introduce a Lagrange multiplier $p \in L^2$ with

$$p = \begin{cases} 0 & \text{if } x \in N \\ -\nabla \cdot (\mu \nabla u) - f & \text{if } x \in A \end{cases}$$

With this definition is also becomes clear that the physical meaning of the Lagrange multiplier is a force. This force acts against the PDE in order to fulfill the constraint.

Then we can re-formulate the strong form as:

Formulation 13.6. *Find* $u : \Omega \to \mathbb{R}$ *and* $p : \Omega \to \mathbb{R}$

$$\begin{aligned} -\nabla \cdot (\mu \nabla u) - p &= f \quad \text{in } \Omega, \\ u &\geq g \quad \text{in } \Omega, \\ p &\geq 0 \quad \text{in } \Omega, \\ p(u-g) &= 0 \quad \text{in } \Omega, \\ u &= u_0 \quad \text{on } \partial\Omega \\ u &= g \quad \text{on } \Gamma \\ \mu \nabla u \cdot n &= \mu \nabla g \cdot n \quad \text{on } \Gamma. \end{aligned}$$

13.3.0.4 Brief introduction to one possibility to treat the obstacle constraint In practice we face the question, how we can realize the obstacle constraint. One possibility (in its basic form not the best!) is penalization, which is a well-known technique in nonlinear programming, e.g., [129, 130]. The idea is to asymptotically fulfill the constraints by including an additional term that acts against the optimization goal if the constraints are violated.

We introduce the penalty parameter $\rho > 0$ (and have $\rho \to \infty$ in mind for later). We have from before: Find $u \in K$:

$$(\mu \nabla u, \nabla(v-u)) \geq (f, v-u) \quad \forall v \in K.$$

A penalized version reads: Find $u_\rho \in H_0^1$:

$$(\mu \nabla u_\rho, \nabla v) - \rho \int_\Omega [g-u]_+ v \, dx = (f, v) \quad \forall v \in H_0^1.$$

Here, $[x]_+ = \max(x, 0)$.

Indeed we have

- $u_\rho \geq g$ yields $0 \geq g - u$. Thus $[g-u]_+ = 0$.
- $u_\rho < g$ yields $0 < g - u$. Thus $[g-u]_+ > 0$.

Remark 13.7. *For large ρ (which are necessary to enforce as well as possible the constraint), the system matrix becomes ill-conditioned and thus the linear system cannot be solved anymore. Here, one has to find a trade-off between sufficiently small and large ρ. An extension is an augmented Lagrangian method or active set methods.*

13.4 Exercise: FEM for the penalized 1D obstacle problem

We derive explicitly the final linear equation system for the 1D situation presented in Section 13.3.0.4:

Formulation 13.8 (1D obstacle problem)**.** *Let $\Omega = (0,1)$ and $f = -1$. The obstacle is given by $g = -0.075$ for all $x \in \Omega$. Find $u : \Omega \to \mathbb{R}$ such that*

$$\begin{aligned} -u'' &\geq f \quad \text{in } \Omega, \\ u &\geq g \quad \text{in } \Omega, \\ [f + u''][g-u] &= 0 \quad \text{in } \Omega, \\ u(0) = u(1) &= 0. \end{aligned}$$

13.4.1 Tasks

We derive in detail the linear equation system for the 1D obstacle problem and adopt the following steps:

1. Write down the 1D obstacle problem in weak formulation (variational inequality).
2. Penalize the inequality constraint using a penalization parameter ρ in order to obtain a variational formulation.
3. Since the problem is nonlinear, employ a Newton or fixed-point method (see Chapter 13) to formulate an iterative scheme for the numerical solution.
4. Discretize in space by choosing linear FEM with $V_h = \{\phi_1, \dots, \phi_n\}$ with $\dim(V_h) < \infty$ and representing u_h as usually done via a linear combination of the basis functions.
5. State the resulting linear equation system $AU = b$ with $U = \{u_1, \dots, u_n\} \in \mathbb{R}^n$
6. Try to justify that a large penalty parameter ρ yields an ill-conditioned system.
7. Make at least a guess (better a detailed justification) how ρ could be chosen in accordance with the spatial discretization parameter h such that ill-conditioning is avoided and we obtain asympotically the correct scheme.
8. Formulate a saddle-point formulation representing the constraint with the help of a Lagrange multiplier p.
9. Propose (in less detail than for the penalized version) an FEM scheme to solve the saddle-point system.

13.5 A variational inequality II: Phase-field fracture

Fracture or damage models allow to simulate discontinuities in solids. A regularized phase-field model reads [96, 139]:

Formulation 13.9. *Find $u : B \to \mathbb{R}^n$:*

$$\begin{aligned} -\nabla \cdot \Big(\big((1-\kappa)\varphi^2 + \kappa \big) \sigma \Big) &= 0 \quad \text{in } B \\ u &= 0 \quad \text{on } \partial B. \end{aligned}$$

The phase-field system consists of three parts: the partial differential equation, the inequality constraint, and a compatibility condition (which is called Rice condition in the presence of fractures [110]): Find $\varphi : B \to [0,1]$ such that

$$\begin{aligned} (1-\kappa)\sigma : e(u)\,\varphi - G_c \varepsilon \Delta \varphi - \frac{G_c}{\varepsilon}(1-\varphi) &\le 0 \quad \text{in } B, \\ \partial_t \varphi &\le 0 \quad \text{in } B, \\ \Big[(1-\kappa)\sigma : e(u)\,\varphi - G_c \varepsilon \Delta \varphi - \frac{G_c}{\varepsilon}(1-\varphi) \Big] \cdot \partial_t \varphi &= 0 \quad \text{in } B, \\ \partial_n \varphi &= 0 \quad \text{on } \partial B. \end{aligned}$$

We explain all sympols in the following:

- ∇*: Nabla operator.*
- $\kappa > 0$*: regularization parameter that system remains solvable when $\varphi \to 0$.*
- φ*: phase-field variable.*
- σ*: stress tensor, e.g., $\sigma = 2\mu e(u) + \lambda tr(e) I$.*
- u*: displacement field.*

- $\mu, \lambda > 0$: *Lamé parameters.*
- $e(u) = \frac{1}{2}(\nabla u + \nabla u^T)$: *strain tensor*
- G_c: *critical energy release rate.*
- ε: *phase-field regularization parameter. Determines the thickness of the transition layer in which* φ *changes from* 0 *to* 1 *and vice versa.*

Remark 13.10. *As in the obstacle problem, we deal with a variational inequality that satisfies a compatiblity condition.*

13.6 Differentiation of nonlinear operators

We have learned about differentiation in Banach spaces in Section 7.4. Examples are:

$$T(u) = u^2$$

then

$$T'_u(u)(h) = 2u \cdot h.$$

Or for semi-linear forms:

$$a(u)(\phi) = (u^2, \phi)$$

we want to differentiate in the first argument:

$$a'_u(u)(h, \phi) = (2u \cdot h, \phi).$$

13.7 Linearization techniques: brief overview

We discuss linearization techniques in the following subsections. The idea is to provide algorithmic frameworks that serve for the implementation. Concerning Newton's method for general problems there is not that much theory; see e.g., [40]. In general one can say that in many problems the theoretical assumptions are not met, but nevertheless Newton's method works well in practice.

Our list is as follows:

- Fixed-point iteration;
- Linearization via time-lagging;
- Newton's method.

As nice online overview is given by Langtangen: http://hplgit.github.io/num-methods-for-PDEs/doc/pub/nonlin/html/slides_nonlin-1.html#nonlin:timediscrete:logistic:Picard

13.8 Fixed-point iteration

13.8.1 Idea

In ODE computations, applying a fixed-point theorem, namely the Banach fixed point theorem, is called a **Picard iteration**. The basic idea is to introduce an iteration using an index k and to linearize the nonlinear terms by taking these terms from the previous iteration $k - 1$.

13.8.2 Example

This is best illustrated in terms of an example. Let

$$-\Delta u + u^2 = f$$

The variational formulation reads:

$$(\nabla u, \nabla \phi) + (u^2, \phi) = (f, \phi) \quad \forall \phi \in V.$$

An iterative scheme is constructed as follows:

Algorithm 13.11 (Fixed-point iterative scheme). *For $k = 1, 2, 3, \ldots$, given u^{k-1}, we seek $u^k \in V$ such that*

$$(\nabla u^k, \nabla \phi) + (u^k u^{k-1}, \phi) = (f, \phi) \quad \forall \phi \in V,$$

until a stopping criterion is fulfilled (choice one out of four):

- *Error criterion:*

$$\|u^k - u^{k-1}\| < TOL, \quad \textit{(absolute)}$$
$$\|u^k - u^{k-1}\| < TOL\|u^k\|, \quad \textit{(relative)}$$

- *Residual criterion:*

$$\|(\nabla u^k, \nabla \phi_i) + ([u^2]^k, \phi_i) - (f, \phi_i)\| < TOL, \quad \textit{(absolute)}$$
$$\|(\nabla u^k, \nabla \phi_i) + ([u^2]^k, \phi_i) - (f, \phi_i)\| < TOL\|(f, \phi_i)\|, \quad \textit{(relative)}$$

  for all $i = 1, \ldots, dim(V_h)$.

Algorithm 13.12 (Fixed-point iterative scheme with relaxation (I)). *Given a damping parameter $\lambda > 0$. For $k = 1, 2, 3, \ldots$, given u^{k-1}, we seek $u^k \in V$ such that*

$$(\nabla u^k, \nabla \phi) + \lambda(u^k u^{k-1}, \phi) = (f, \phi) \quad \forall \phi \in V,$$

until a stopping criterion is fulfilled.

Algorithm 13.13 (Fixed-point iterative scheme with relaxation (II)). *For $k = 1, 2, 3, \ldots$, given u^{k-1}:*

1. *We seek $\hat{u}^k \in V$*
$$(\nabla \hat{u}^k, \nabla \phi) + (\hat{u}^k u^{k-1}, \phi) = (f, \phi) \quad \forall \phi \in V,$$
2. *Set $u^k := \lambda \hat{u}^k + (1 - \lambda) u^{k-1}$ with $\lambda \in (0, 1]$;*
3. *Until a stopping criterion is fulfilled.*

We briefly provide some details on the linearized part. We have

$$(u^k u^{k-1}, \phi) = \int_\Omega u^{k-1}(x) u^k(x) \cdot \phi(x)\, dx.$$

Using the FEM representation of u^k, we obtain

$$u^k = \sum_{j=1}^{M} a_j \phi_j,$$

where $M = dim(V_h) = \{\phi_1, \ldots, \phi_M\}$ and a_j are as usual the unknown solution coefficients. Plugging this in, we obtain

$$\sum_j a_j(u^{k-1}\phi_j, \phi_i) = \sum_j a_j \int_\Omega u^{k-1}(x)\phi_j(x) \cdot \phi_i(x)\, dx.$$

We recall that $u^{k-1} \in V_h$ at the old iteration step $k-1$. Working with $V_h := V_h^{(1)}$ means that u^{k-1} is a linear function on the element K_l. In principle the previous integral can be evaluated by building the anti-derivative. The more convenient way is to use numerical quadrature. To this end, we obtain on the element K_l over our triangulation the following integral:

$$\int_{K_l} u^{k-1}(x)\phi_j(x) \cdot \phi_i(x)\, dx \approx \sum_{q=1}^{N_q} u^{k-1}(x_q)\phi_j(x_q) \cdot \phi_i(x_q)\omega(x_q)$$

where N_q is the number of quadrature points and where x_q are the quadrature points on the elment K_l and $\omega(x_q)$ are the quadrature weights. Specifically, $u^{k-1}(x_q)$ is the evaluation of the previous FEM $k-1$th solution at the point x_q.

Algorithm 13.14 (Fixed-point iterative scheme; stronger linearization)**.** *Given a damping parameter $\lambda > 0$. A stronger linearization would be as follows: For $k = 1, 2, 3, \ldots$, given u^{k-1}, we seek $u^k \in V$ such that*

$$(\nabla u^k, \nabla\phi) + \lambda((u^{k-1})^2, \phi) = (f, \phi) \quad \forall \phi \in V,$$

which means that the entire nonlinearity goes on the right hand side:

$$(\nabla u^k, \nabla\phi) = (f, \phi) - ((u^{k-1})^2, \phi) \quad \forall \phi \in V.$$

Remark 13.15. *For time-dependent PDEs, a common (simple) linearization can be*

$$(u^2, \phi) \to (uu^{n-1}, \phi)$$

where $u^{n-1} := u(t^{n-1})$ is the previous time step solution. In this case, no additional fixed-point iteration needs to be constructed because the linearization is part of the temporal discretization. An even stronger linearization would be

$$(u^2, \phi) \to ((u^{n-1})^2, \phi).$$

For instance, for the nonlinear parabolic PDE

$$\partial_t u - \Delta u + u^2 = f$$

we obtain in the semi-time-discrete (implicit Euler) variational forms:

$$\text{Find } u^n \in V: \quad (u^n, \phi) + k(\nabla u^n, \nabla\phi) + k(u^n u^{n-1}, \phi) = k(f^n, \phi) - (u^{n-1}, \phi) \quad \forall \phi \in V,$$

and

$$\text{Find } u^n \in V: \quad (u^n, \phi) + k(\nabla u^n, \nabla\phi) = k(f^n, \phi) - k((u^{n-1})^2, \phi) - (u^{n-1}, \phi) \quad \forall \phi \in V,$$

respectively.

13.8.3 Example of a 1D nonlinear equation

Given the nonlinear problem

$$\begin{aligned} -(uu')' &= f \quad \text{in } \Omega \\ u(0) &= u(1) = 0. \end{aligned}$$

Thus we have a nonlinear coefficient $a(u) = u$ in front of the highest order derivative. The weak formulation reads: Find $u \in H_0^1(\Omega)$ such that

$$(uu', \varphi') = (f, \varphi) \quad \forall \varphi \in H_0^1(\Omega).$$

A fixed-point linearization would be: given some initial guess u_0, iterate for $k = 1, 2, 3, \ldots$ such that in weak formulation:

$$(u_{k-1}u_k', \varphi') = (f, \varphi)$$

The nonlinear coefficient u_{k-1} is taken at the previous iteration step $k-1$. Now we are in the game and can take as discrete representation in $V_h = \{\varphi_1, \ldots, \varphi_n\}$:

$$u_k = \sum_{j=1}^{n} a_j \varphi_j.$$

The solution coefficients are denoted by a_j (we did not take u_j in order to avoid any confusion with the iteration index). Then:

$$\sum_{j=1}^{n} a_j (u_{k-1}\varphi_j', \varphi_i') = (f, \varphi_i) \quad \text{for } i = 1, \ldots, n$$

Therefore, we obtain similar results to Section 8.4, with the modification that we need to evaluate the previous FEM solution u_{k-1} in each integral $(u_{k-1}\varphi_j', \varphi_i')$.

13.9 Linearization via time-lagging (example: NSE)

For time-dependent PDEs, a common linearization can be

$$(u^2, \phi) \to (uu^{n-1}, \phi)$$

where $u^{n-1} := u(t^{n-1})$ is the previous time step solution. In this case, no additional fixed-point iteration needs to be constructed.

Let us know briefly explain this idea to Navier-Stokes to point an important property. Setting $N(v) := v \cdot \nabla v - \nabla \cdot (\nabla v + \nabla v^T)$, we write

$$\partial_t v + N(v) + \nabla p = f, \quad \nabla \cdot v = 0$$

be given.

13.9.1 Stokes linearization

Using the Stokes linearization (for small Reynolds numbers), the nonlinearity can be treated fully explicitely:

$$v + k\nabla p = v^{n-1} - kN(v^{n-1})k\theta f + k(1-\theta)f^{n-1}, \quad \nabla \cdot v = 0.$$

This simplifies the problem as it is now linear and in each step, the symmetric and positive Stokes-operator must be inverted.

13.9.2 Oseen linearization

For higher Reynold's numbers, the so-called Oseen linearization can be consulted:

$$v + k\theta\tilde{N}(v) + k\nabla p = v^{n-1} - k(1-\theta)\tilde{N}(v^{n-1})k\theta f + k(1-\theta)f^{n-1}, \quad \nabla \cdot v = 0,$$

where

$$\tilde{N}(v) = \tilde{v} \cdot \nabla v,$$

where, for instance a constant extrapolation $\tilde{v} = v^{n-1}$ or linear extrapolation $\tilde{v} = 2v^{n-1} - v^{n-2}$ can be employed; see also the next section. Here, at each step, a (linear) nonsymmetric diffusion-transport operator must be inverted.

13.10 Newton's method in $\mathbb{R}$ - the Newton-Raphson method

Let $f \in C^1[a,b]$ with at least one point $f(x) = 0$, and $x_0 \in [a,b]$ be a so-called initial guess. The task is to find $x \in \mathbb{R}$ such that

$$f(x) = 0.$$

In most cases it is impossible to calculate x explicitly. Rather we construct a sequence of iterates $(x_k)_{k\in\mathbb{R}}$ and hopefully reach at some point

$$|f(x_k)| < TOL, \quad \text{where } TOL \text{ is small, e.g., } TOL = 10^{-10}.$$

What is true for all Newton derivations in the literature is that one has to start with a Taylor expansion. In our lecture we do this as follows. Let us assume that we are at x_k and can evaluate $f(x_k)$. Now we want to compute this next iterate x_{k+1} with the unknown value $f(x_{k+1})$. Taylor gives us:

$$f(x_{k+1}) = f(x_k) + f'(x_k)(x_{k+1} - x_k) + o(x_{k+1} - x_k)^2$$

We assume that $f(x_{k+1}) = 0$ (or very close to zero $f(x_{k+1}) \approx 0$). Then, x_{k+1} is the sought root and neglecting the higher-order terms we obtain:

$$0 = f(x_k) + f'(x_k)(x_{k+1} - x_k).$$

Thus:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, 2, \ldots. \tag{280}$$

This iteration is possible as long as $f'(x_k) \neq 0$.

Remark 13.16 (Relation to Section 10.3)**.** *We see that Newton's method can be written as*

$$x_{k+1} = x_k + d_k, \quad k = 0, 1, 2, \ldots,$$

where the search direction is

$$d_k = -\frac{f(x_k)}{f'(x_k)}.$$

The iteration (280) terminates if a stopping criterion

$$\frac{|x_{k+1} - x_k|}{|x_k|} < TOL, \quad \text{or} \quad |x_{k+1} - x_k| < TOL, \tag{281}$$

or

$$|f(x_{k+1})| < TOL \tag{282}$$

is fulfilled. All these TOL do not need to be the same, but sufficiently small and larger than machine precision.

Remark 13.17. *Newton's method belongs to fix-point iteration schemes with the iteraction function:*

$$F(x) := x - \frac{f(x)}{f'(x)}. \tag{283}$$

For a fixed point $\hat{x} = F(\hat{x})$ it holds: $f(\hat{x}) = 0$. Compare again to Section 10.3.

The main results is given by:

Theorem 13.18 (Newton's method)**.** *The function $f \in C^2[a,b]$ has a root $\hat{x}$ in the interval $[a,b]$ and*

$$m := \min_{a\le x\le b} |f'(x)| > 0, \quad M := \max_{a\le x\le b} |f''(x)|.$$

Let $\rho > 0$ such that

$$q := \frac{M}{2m}\rho < 1, \quad K_\rho(\hat{x}) := \{x \in \mathbb{R} : |x - \hat{x}| \le \rho\} \subset [a,b].$$

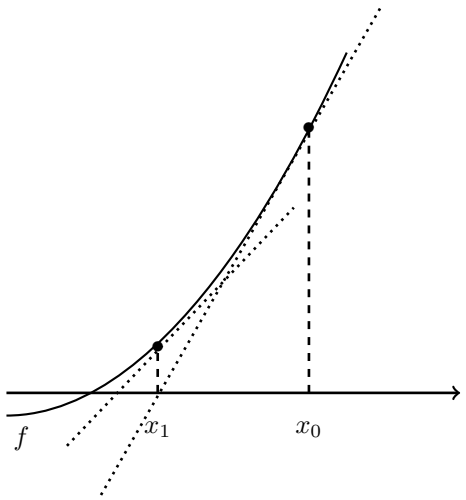


Figure 63: Geometrical interpretation of Newton's method.

Then, for any starting point $x_0 \in K_\rho(\hat{x})$, the sequence of iterations $x_k \in K_\rho(\hat{x})$ converges to the root $\hat{x}$. Furthermore, we have the a priori estimate

$$|x_k - \hat{x}| \leq \frac{2m}{M} q^{2^k}, \quad k \in \mathbb{N},$$

and a posteriori estimate

$$|x_k - \hat{x}| \leq \frac{1}{m}|f(x_k)| \leq \frac{M}{2m}|x_k - x_{k+1}|^2, \quad k \in \mathbb{N}.$$

Often, Newton's method is formulated in terms of a defect-correction scheme.

Definition 13.19 (Defect)**.** *Let $\tilde{x} \in \mathbb{R}$ an approximation of the solution $f(x) = y$. The defect (or similarly the residual) is defined as*

$$d(\tilde{x}) = y - f(\tilde{x}).$$

Definition 13.20 (Newton's method as defect-correction scheme)**.**

$$\begin{aligned} f'(x_k)\delta x = d_k, \quad d_k &:= y - f(x_k), \\ x_{k+1} = x_k + \delta x, \quad k &= 0, 1, 2, \ldots. \end{aligned}$$

The iteration is finished with the same stopping criterion as for the classical scheme. To compute the update δx we need to invert $f'(x_k)$:

$$\delta x = (f'(x_k))^{-1} d_k.$$

This step seems trivial but is the most critical one if we deal with problems in $\mathbb{R}^n$ with $n > 1$ or in function spaces. Because here, the derivative becomes a matrix. Therefore, the problem results in solving a linear equation system of the type $A\delta x = b$ and computing the inverse matrix A^{-1} is an expensive operation. $\diamond$

Remark 13.21. *This previous forms of Newton's method are already very close to the schemes that are used in research. One simply extends from $\mathbb{R}^1$ to higher dimensional cases such as nonlinear PDEs or optimization. The 'only' aspects that are however big research topics are the choice of*

- *good initial Newton guesses;*
- *globalization techniques.*

Two very good books on these topics, including further materials as well, are [40, 97].

13.10.1 Newton's method: overview. Going from $\mathbb{R}$ to Banach spaces

Overview:

- Newton-Raphson (1D), find $x \in \mathbb{R}$ via iterating $k = 0, 1, 2, \ldots$ such that $x_k \approx x$ via:

$$\begin{aligned} \text{Find } \delta x \in \mathbb{R}: \quad & f'(x_k)\delta x = -f(x_k), \\ \text{Update:} \quad & x_{k+1} = x_k + \delta x. \end{aligned}$$

- Newton in $\mathbb{R}^n$, find $x \in \mathbb{R}^n$ via iterating $k = 0, 1, 2, \ldots$ such that $x_k \approx x$ via:

$$\begin{aligned} \text{Find } \delta x \in \mathbb{R}^n: \quad & F'(x_k)\delta x = -F(x_k), \\ \text{Update:} \quad & x_{k+1} = x_k + \delta x. \end{aligned}$$

  Here we need to solve a linear equation system to compute the update $\delta x \in \mathbb{R}^n$.

- Banach spaces, find $u \in V$, with $dim(V) = \infty$, via iterating $k = 0, 1, 2, \ldots$ such that $u_k \approx u$ via:

$$\begin{aligned} \text{Find } \delta u \in V: \quad & F'(u_k)\delta u = -F(u_k), \\ \text{Update:} \quad & u_{k+1} = u_k + \delta u. \end{aligned}$$

  Such a problem needs to be discretized and results again in solving a linear equation system in the defect step.

- Banach spaces, applied to variational formulations, find $u \in V$, with $dim(V) = \infty$, via iterating $k = 0, 1, 2, \ldots$ such that $u_k \approx u$ via:

$$\begin{aligned} \text{Find } \delta u \in V: \quad & a'(u_k)(\delta u, \phi) = -a(u_k)(\phi), \\ \text{Update:} \quad & u_{k+1} = u_k + \delta u. \end{aligned}$$

  As before, the infinite-dimensional problem is discretized resulting in solving a linear equation system in the defect step.

13.10.2 A basic algorithm for a residual-based Newton method

In this type of methods, the main criterion is a decrease of the residual in each step:

Algorithm 13.22 (Residual-based Newton's method). *Given an initial guess x_0. Iterate for $k = 0, 1, 2, \ldots$ such that*

$$\begin{aligned} \textit{Find } \delta x \in \mathbb{R}^n: \quad & F'(x_k)\delta x_k = -F(x_k), \\ \textit{Update:} \quad & x_{k+1} = x_k + \lambda_k \delta x_k, \end{aligned}$$

with $\lambda_k \in (0, 1]$ (see the next sections how λ_k can be determined). A full Newton step corresponds to $\lambda_k = 1$. The criterion for convergence is the contraction of the residuals measured in terms of a discrete vector norm:

$$\|F(x_{k+1})\| < \|F(x_k)\|.$$

In order to save some computational cost, close to the solution x^*, intermediate simplified Newton steps can be used. In the case of $\lambda_k = 1$ we observe

$$\theta_k = \frac{\|F(x_{k+1})\|}{\|F(x_k)\|} < 1.$$

If $\theta_k < \theta_{max}$, e.g., $\theta_{max} = 0.1$, then the old Jacobian $F'(x_k)$ is kept and used again in the next step $k + 1$. Otherwise, if $\theta_k > \theta_{max}$, the Jacobian will be assembled. Finally the stopping criterion is one of the following (relative preferred!):

$$\begin{aligned} &\|F(x_{k+1})\| \leq TOL_N \quad \textit{(absolute)} \\ &\|F(x_{k+1})\| \leq TOL_N \|F(x_0)\| \quad \textit{(relative)} \end{aligned}$$

If fulfilled, set $x^* := x_{k+1}$ and the (approximate) root x^* of the problem $F(x) = 0$ is found.

Remark 13.23 (C++ Newton implementation). *A C++ implementation can be found of such a basic Newton scheme can be found in [122].*

13.11 Inexact Newton

For large scale nonlinear PDEs, the inner linear systems are usually not solved with a direct method (e.g., LU), but with iterative methods (CG, GMRES, multigrid). Using such iterative methods yields an **inexact Newton scheme**. Here, we deal with two iterations:

- The outer nonlinear Newton iteration.
- The inner linear iteration.

Algorithm 13.24 (Inexact Newton method)**.** *Given an initial guess x_0. Iterate for $k = 0, 1, 2, \dots$ such that*

$$\begin{aligned} \textit{Formulate } \delta x \in \mathbb{R}^n: \quad & F'(x_k)\delta x_k = -F(x_k), \\ \textit{Solve with an iterative linear method } & F'(x_k)\delta x_k^i = -F(x_k), \\ \textit{Check linear stopping criterium} & \|\delta x_k^i - \delta x_k^{i-1}\| < TOL_{Lin}, \\ \textit{Update: } & x_{k+1} = x_k + \lambda_k \delta x_k^i, \\ \textit{Check nonlinear stopping criterium, e.g., } & \|x_{k+1} - x_k\| < TOL_{New} \end{aligned}$$

with $\lambda_k \in (0, 1]$.

13.12 Newton's method for variational formulations

In this section, we describe in detail an Newton method for solving nonlinear nonstationary PDE problems in variational form. That is we have the following indices:

- h for spatial discretization.
- n for the current time step solution.
- j Newton iteration.
- l (optional): line search iterations.

Therefore, we deal with the following problem on the discrete level:

$$a(u_h^n)(\phi) = l(\phi) \quad \forall \phi \in V_h,$$

which is solved with a Newton-like method. We can express this relation in terms of the defect:

$$d := l(\phi) - a(u_h^n)(\phi) = 0 \quad \forall \phi \in V_h.$$

Algorithm 13.25 (Basic Newton method as defect-correction problem)**.** *Given an initial Newton guess $u_h^{n,0} \in V_h$, find for $j = 0, 1, 2, \dots$ the update $\delta u_h^n \in V_h$ of the linear defect-correction problem*

$$a'(u_h^{n,j})(\delta u_h^n, \phi) = -a(u_h^{n,j})(\phi) + l(\phi), \tag{284}$$

$$u_h^{n,j+1} = u_h^{n,j} + \lambda \delta u_h^n, \tag{285}$$

$$\textit{Check } \| - a(u_h^{n,j})(\phi) + l(\phi)\| < TOL_N, \tag{286}$$

$$\textit{or check } \| - a(u_h^{n,j})(\phi) + l(\phi)\| < TOL_N \| - a(u_h^{n,0})(\phi) + l(\phi)\|, \tag{287}$$

with a line search damping parameter $\lambda \in (0, 1]$ and a Newton tolerance TOL_N, e.g., $TOL_N = 10^{-10}$. For $\lambda = 1$, we deal with a full Newton step. Adapting λ and choosing $0 < \lambda < 1$ helps to achieve convergence of Newton's method for certain nonlinear problems when a full Newton step does not work.

Remark 13.26 (Initial Newton guess for time-dependent problems)**.** *In time-dependent problems the 'best' initial Newton guess is the previous time step solution; namely*

$$u_h^{n,0} := u_h^{n-1}.$$

Remark 13.27 (Initial Newton guess in general)**.** *When possible the solution should be obtained in nested iterations. We start on a coarse mesh with mesh size h_0 and here $u^0_{h_0} = 0$ when nothing better is available. Then, we compute the solution u_{h_0}. After Newton converged, we project u_{h_0} on the next mesh and use $u^0_{h_1} := u_{h_0}$ as initial Newton guess and repeat the entire procedure.*

Remark 13.28 (Dirichlet boundary conditions)**.** *In Newton's method, non-homogeneous Dirichlet boundary conditions are only prescribed on the initial guess $u_h^{n,0}$ and in all further updates non-homogeneous Dirichlet conditions are replaced by homoegeneous Dirichlet conditions. The reasons is that we only need to prescribe these conditions once per Newton iteration and not several times, which would lead to wrong solution.*

In the following we present a slightly more sophisticated algorithm with two additional features:

- backtracking line search to choose λ;
- simplified Newton steps in order to avoid too many assemblings of the Jacobian matrix $a'(u_h^{n,j})(\delta u_h^n, \phi)$.

The second point is a trade-off: the less assemblings we do, the more Newton's performance deteriorates and quadratic convergence gets lost resulting in a fixed-point-like scheme in which many more nonlinear iterations j are required to achieve the tolerance. On the other hand, when we do not construct the Jacobian, we save the assembling and can keep working with the previous matrix. For details, we refer for instance to [40].

Algorithm 13.29 (Backtracking line search)**.** *A simple strategy is to modify the update step in* (284) *as follows: For given $\lambda \in (0,1)$ determine the minimal $l^* \in \mathbb{N}$ via $l = 0, 1, \ldots, N_l$, such that*

$$\|R(u_{h,l}^{n,j+1})\|_\infty < \|R(u_{h,l}^{n,j})\|_\infty,$$
$$u_{h,l}^{n,j+1} = u_h^{n,j} + \lambda^l \delta u_h^n.$$

For the minimal l, we set

$$u_h^{n,j+1} := u_{h,l^*}^{n,j+1}.$$

In this context, the nonlinear residual $R(\cdot)$ is defined as

$$R(u_h^{n,j})(\phi_i) := a(u_h^{n,j})(\phi_i) - l(\phi_i) \quad \forall \phi_i \in V_h,$$

and

$$\|R(u_h^{n,j})\|_\infty := \max_{i \in [1, dim(V_h)]} \{R(u_h^{n,j})(\phi_i)\}$$

The final full Newton algorithm based on a contraction of the residuals reads:

Algorithm 13.30 (Residual-based Newton's method with backtracking line search and simplified Newton steps)**.** *Given an initial Newton guess $u_h^{n,0} \in V_h$. For the iteration steps $j = 0, 1, 2, 3, \ldots$:*

1. *Find $\delta u_h^n \in V_h$ such that*

$$a'(u_h^{n,j})(\delta u_h^n, \phi) = -a(u_h^{n,j})(\phi) + l(\phi) \quad \forall \phi \in V_h, \tag{288}$$
$$u_h^{n,j+1} = u_h^{n,j} + \lambda_j \delta u_h^n, \tag{289}$$

   for $\lambda_j = 1$.

2. *The criterion for convergence is the contraction of the residuals:*

$$\|R(u_h^{n,j+1})\|_\infty < \|R(u_h^{n,j})\|_\infty. \tag{290}$$

3. *If* (290) *is violated, re-compute in* (288) *$u_{h,l}^{n,j+1}$ by choosing $\lambda_j^l = 0.5$, and compute for $l = 1, ..., l_M$ (e.g. $l_M = 5$) a new solution*

$$u_h^{n,j+1} = u_h^{n,j} + \lambda_j^l \delta u_h^n,$$

   until (290) *is fulfilled for a $l^* < l_M$ or l_M is reached. In the latter case, no convergence is obtained and the program aborts.*

4. In case of $l^* < l_M$ we check next the stopping criterion:

$$\|R(u_h^{n,j+1})\|_\infty \leq TOL_N, \quad \textit{(absolute)}$$
$$\|R(u_h^{n,j+1})\|_\infty \leq TOL_N \|R(u_h^{n,0})\|_\infty, \quad \textit{(relative)}$$

If this is criterion is fulfilled, set $u_h^n := u_h^{n,j+1}$. Else, we increment $k \to k+1$ and goto Step 1.

Remark 13.31 (On using simplified-Newton steps)**.** *Usually, when the Newton reduction rate*

$$\theta_k = \frac{\|R(u_h^{n,j+1})\|}{\|R(u_h^{n,j})\|},$$

was sufficiently good, e.g., $\theta_k \leq \theta_{max} < 1$ (where e.g. $\theta_{max} \approx 0.1$), a common strategy is to work with the 'old' Jacobian matrix, but with a new right hand side.

13.13 Continuing Section 13.8.2, now with Newton

Given from Section 13.8.2:

$$(\nabla u, \nabla \phi) + (u^2, \phi) = (f, \phi) \quad \forall \phi \in V$$

Let us formulate now a Newton scheme: Find $\delta u \in V$ such that

$$(\nabla \delta u, \nabla \phi) + (2u^k \delta u, \phi) = -(\nabla u^k, \nabla \phi) - ((u^k)^2, \phi) + (f, \phi) \quad \phi \in V,$$
$$u^{k+1} = u^k + \delta u.$$

13.14 Continuing Section 13.8.3, now with Newton

Taking the problem from Section 13.8.3, we first formulate the semi-linear form (we could have done this therein as well):

$$a(u)(\varphi) = (uu', \varphi') - (f, \varphi).$$

Thus we want to solve:

$$a(u)(\varphi) = 0.$$

For this, we need (see e.g., Section 13.10.1):

$$\text{Find } \delta u \in V: \quad a'(u_k)(\delta u, \varphi) = -a(u_k)(\varphi),$$
$$\text{Update:} \qquad u_{k+1} = u_k + \delta u.$$

Thus, we need to clarify the meanings of $u_k, \delta u, \varphi$:

- u_k is the previous iterate
- δu is our unknown and therefore the trial function
- φ as usual the test function

As derivative for $a(u)(\varphi)$, we obtain with the help of Section 7.4:

$$a'(u)(\delta u, \varphi) = (\delta u u' + u \delta u', \varphi')$$

Then, the defect step of Newton's method at iteration k reads for the unknown δu:

$$(\delta u u_k' + u_k \delta u', \varphi') = -(u_k u_k', \varphi') + (f, \varphi)$$

We need to represent δu as FEM function:

$$\delta u = \sum_{j=1}^{n} a_j \varphi_j.$$

Since the left-hand-side is linear and the right-hand-side only contains given data, we obtain:

$$\sum_{j=1}^{n} a_j(\varphi_j u_k' + u_k\varphi_j', \varphi_i') = -(u_k u_k', \varphi_i') + (f, \varphi_i)$$

Similar to Section 13.8.3, we are now in the game to known topics; see Section 8.4. In a bit more detail, we need to assemble a matrix in which φ and φ arise, and a standard stiffness matrix with coefficients depending on the previous solution:

$$\sum_{j=1}^{n} a_j \Big((\varphi_j u_k', \varphi_i') + (u_k\varphi_j', \varphi_i')\Big) = -(u_k u_k', \varphi_i') + (f, \varphi_i)$$

Thus:

$$AU = B$$

with

$$U = (a_j)_j \in \mathbb{R}^n, \quad A = ((\varphi_j u_k', \varphi_i') + (u_k\varphi_j', \varphi_i'))_{ij} \in \mathbb{R}^{n\times n}, \quad B = (-(u_k u_k', \varphi_i') + (f, \varphi_i))_i \in \mathbb{R}^n$$

13.15 The regularized p-Laplacian

As previously characterized, the p-Laplacian is an example of a quasi-linear PDE. In this section, we state the complete problem, present an error analysis and then a numerical test. In this example, we confirm the theoretical error analysis and also discuss iteration numbers of the nonlinear and linear solvers. In the latter one, we use a geometric multigrid method as preconditioner.

13.15.1 Problem statement

We assume Ω being a bounded polygonal domain in $\mathbb{R}^d$, with $d = 2$ and $\Gamma_D = \partial\Omega$. We consider the following scalar p-type problem

$$-\mathrm{div}\mathbf{A}(\nabla u) = f \quad in\ \Omega, \qquad u = u_D \quad on\ \Gamma_D, \tag{291}$$

where $f : \Omega \to \mathbb{R}$ and $u_D : \Gamma_D \to \mathbb{R}$ are given smooth functions. The operator $\mathbf{A}(\nabla u) : \mathbb{R}^2 \to \mathbb{R}^2$ has the following p-power law form

$$\mathbf{A}(\nabla u) = (\varepsilon^2 + |\nabla u|^2)^{\frac{p-2}{2}}\nabla u, \tag{292}$$

where $p \in (1, \infty)$ and $\varepsilon > 0$ are model parameters and $|\cdot|^2 = (\cdot, \cdot)$. The function $a(\nabla u) = (\varepsilon^2 + |\nabla u|^2)^{\frac{p-2}{2}}$ is the diffusivity term of (291). As we can observe by (292), the nonlinear nature of the problem is due to the appearance of $|\nabla u|$ in the diffusivity function and this poses numerical challenges. We introduce the closely related function $\mathbf{F} : \mathbb{R}^2 \to \mathbb{R}^2$ to the operator $\mathbf{A}$ by

$$\mathbf{F}(\mathbf{a}) = (\varepsilon^2 + |\mathbf{a}|^2)^{\frac{p-2}{4}}\mathbf{a}. \tag{293}$$

For the mathematical setting, let $1 \le p \le \infty$ be fixed and l be a non-negative integer. As usual, $L^p(\Omega)$ denotes Lebesgue spaces for which $\int_\Omega |u(x)|^p\,dx < \infty$, endowed with the norm $\|u\|_{L^p(\Omega)} = \big(\int_\Omega |u(x)|^p\,dx\big)^{\frac{1}{p}}$, and $W^{l,p}(\Omega)$ is the Sobolev space, which consists of the functions $\phi : \Omega \to \mathbb{R}$ such that their weak derivatives $D^\alpha\phi$ with $|\alpha| \le l$ belong to $L^p(\Omega)$. If $\phi \in W^{l,p}(\Omega)$, then its norm is defined by

$$\|\phi\|_{W^{l,p}(\Omega)} = \Big(\sum_{0\le|\alpha|\le l} \|D^\alpha\phi\|^p_{L^p(\Omega)}\Big)^{\frac{1}{p}} \quad \text{and} \quad \|\phi\|_{W^{l,\infty}(\Omega)} = \max_{0\le|\alpha|\le l} \|D^\alpha\phi\|_{L^\infty(\Omega)},$$

for $1 \le p < \infty$ and $p = \infty$, respectively. We refer the reader to [1] for more details about Sobolev spaces. Furthermore, we define the spaces

$$W^{l,p}_D := \{u \in W^{l,p}(\Omega) : u|_{\partial\Omega} = u_D\}, \quad \text{and} \quad W^{l,p}_0 := \{u \in W^{l,p}(\Omega) : u|_{\partial\Omega} = 0\}. \tag{294}$$

In what follows, positive constants c and C appearing in the inequalities are generic constants which do not depend on the mesh-size h. We indicate on what may the constants depend for a better understanding of the proofs. Frequently, we will write $a \sim b$ meaning that $ca \leq b \leq Ca$, with c and C independent of the mesh size.

The weak formulation for (291) reads as follows:

Formulation 13.32. *Find $u \in W_D^{1,p}$ such that*

$$B(u,\phi) = l_f(\phi),\ \forall \phi \in W_0^{1,p}(\Omega), \quad \text{where } B(u,\phi) = \int_\Omega \mathbf{A}(\nabla u)\cdot\nabla\phi\,dx, \quad \text{and } l_f(\phi) = \int_\Omega f\phi\,dx. \tag{295a}$$

Depending on the form of $\mathbf{A}$ and on the range of p, the well-posedness has been examined by means of monotone operators in several works, see e.g., [33, 35].

Problem (295) is equivalent to the minimization problem:

Formulation 13.33.

$$\text{Find } \ u \in W_D^{1,p} \text{ such that } J(u) \leq J(\phi),\ \forall \phi \in W_D^{1,p}, \tag{296}$$

where $J : W_D^{1,p} \to \mathbb{R}$ is defined by

$$J(\phi) = \frac{1}{p}\int_\Omega (\varepsilon^2 + |\nabla\phi|^2)^{\frac{p}{2}}\,dx - \int_\Omega f\phi\,dx. \tag{297}$$

Furthermore, one can show that the Gateaux derivative of B is given by

$$B'(u)(v,w) = \int_\Omega (\varepsilon^2 + |\nabla u|^2)^{\frac{p-2}{2}}\nabla v\cdot\nabla w\,dx \tag{298}$$

$$+(p-2)\int_\Omega (\varepsilon^2 + |\nabla u|^2)^{\frac{p-4}{2}}(\nabla u\cdot\nabla v)(\nabla u\cdot\nabla w)\,dx, \quad \text{for} \quad u,v,w \in W_D^{1,p}. \tag{299}$$

Hypothesis 13.34. *Let $l \geq 2$ be an integer and let $p \in (1,\infty)$ and $d = 2$. We assume that the solution u of (295) belongs to $V := W_D^{l,p}(\Omega)$, where either $(l-1)p > d$ and $p > 1$ or $(l-1)p < d$ and $p > \frac{2d}{d+1}$. Further, we assume that $\nabla\mathbf{F}(\nabla u) \in L^2(\Omega)$.*

Theorem 13.35. *Let $u \in V$ be the solution of (295) under the Assumption 13.34, and let $u_h \in V_{D,h}^{(k)}$ be the solution of the discrete p-Laplacian. Then, there exist $C \geq 0$, independent of the grid size h, such that*

$$\int_\Omega |\mathbf{F}(\nabla u) - \mathbf{F}(\nabla u_h)|^2\,dx \leq Ch^{2(l-1)}\|u\|^2_{W^{l,p}(\Omega)}. \tag{300}$$

Proof. See [128]. □

13.15.2 Augmented Lagrangian techniques as one option for the numerical solution

In the following two paragraphs, we transform the original problem (126) into a saddle-point problem using augmented Lagrangian techniques. Let q be the conjugate exponent of p, that is $\frac{1}{p} + \frac{1}{q} = 1$, and let us define the space $W \subset W_D^{1,p} \times (L^p(\Omega))^2$ by

$$W = \{(v,\mathbf{q}) | (v,\mathbf{q}) \in W_D^{1,p} \times (L^p(\Omega))^2 : \nabla v - \mathbf{q} = 0\}. \tag{301}$$

Following [59], we introduce the augmented Lagrangian $\mathcal{L}_{\widehat{r}}$ defined, for $\widehat{r} > 0$, by

$$\mathcal{L}_{\widehat{r}}(v,\mathbf{q},\lambda) \ = \ \frac{1}{p}\int_\Omega (\varepsilon^2 \ + \ |\mathbf{q}|^2)^{\frac{p}{2}}\,dx \ - \ \int_\Omega fv\,dx \ + \ \frac{\widehat{r}}{2}\int_\Omega |\nabla v \ - \ \mathbf{q}|^2\,dx \ + \ \int_\Omega \lambda \ \cdot \ (\nabla v \ - \ \mathbf{q})\,dx, \tag{302}$$

and the following associated to (302) saddle-point problem:

$$\text{Find}\{u,\mathbf{q},\lambda\} \in W_D^{1,p} \times (L^p(\Omega))^2 \times (L^q(\Omega))^2 \text{ such that} \tag{303a}$$

$$\mathcal{L}_{\widehat{r}}(u,\mathbf{q},\mu) \leq \mathcal{L}_{\widehat{r}}(u,\mathbf{q},\lambda) \leq \mathcal{L}_{\widehat{r}}(v,\mathbf{w},\lambda), \tag{303b}$$

$$\forall\{v,\mathbf{w},\mu\} \in W_D^{1,p} \times (L^p(\Omega))^2 \times (L^q(\Omega))^2. \tag{303c}$$

Here the Lagrangian has very similar properties to the Lagrangian defined in Section 9.6.

13.15.3 Numerical test p-Laplacian w. Newton-CG nonlinear/linear solver w. GMG preconditioning

We take example No. 2 from [128] and implement as linear solver the CG method with geometric multigrid preconditioning. The solver configurations are as in Section 10.8, namely $TOL_{Lin} = 1e-12$. For the Newton solver, we choose the tolerance $TOL_{New} = 1e-10$. A plot of the solution is presented in Figure 13.15.3.

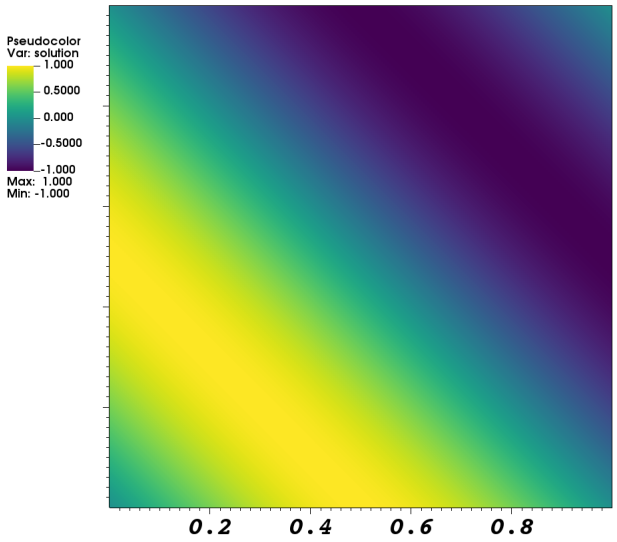


Figure 64: Solution of p-Laplace. Configuration taken from [128], Example No. 2.

We obtain as numerical results:

```
FE degree eps p TOL (LinSolve) TOL(Newton)
============================================================
1 0.1 1.01 1e-12 1e-10
------------------------------------------------------------
Cells   DoFs    h               F-norm err      ConvRate        Min/MaxLinIter  Newton iter
=============================================================================================
4096    4225    2.20971e-02     5.43340e-02     1.04683e+00     18/28           5
16384   16641   1.10485e-02     2.03360e-02     1.41782e+00     27/37           5
65536   66049   5.52427e-03     1.01906e-02     9.96804e-01     25/33           4
262144  263169  2.76214e-03     5.09674e-03     9.99587e-01     25/30           3
1048576 1050625 1.38107e-03     2.54855e-03     9.99899e-01     24/28           3
4194304 4198401 6.90534e-04     1.27430e-03     9.99975e-01     23/25           3
=============================================================================================
```

Observations:

- The F norm is close to 1 as to be expected from the theory stated above and proven in [128]
- The number of linear iterations is (nearly) mesh-independent and asymptotically stable. Therefore, the geometric multigrid preconditioner works very well.
- The number of nonlinear Newton iterations is as well mesh-independent and asymptotically stable

13.16 Temporal discretization of non-linear time-dependent problems

In the domain Ω and the time interval $I = (0, T)$, we consider the abstract problem

$$\overline{A}(U)(\Psi) = \overline{F}(\Psi)$$

where

$$\overline{A}(U)(\Psi) = \int_0^T A(U)(\Psi)\, dt, \quad \overline{F}(\Psi) = \int_0^T F(\Psi)\, dt.$$

The abstract problem can either be treated by a full time-space Galerkin formulation, which was investigated previously for fluid problems in Besier et al. [18, 19, 119]. Alternatively, the Rothe method or the method of lines can be used in cases where temporal and spatial discretization are treated separtetely.

For reasons explained in Chapter 12, we follow the Rothe method. After semi-discretization in time, we obtain a sequence of generalized steady-state fluid-structure interaction problems that are completed by appropriate boundary values at every time step. Let us now go into detail and let

$$I = \{0\} \cup I_1 \cup \ldots \cup I_N$$

be a partition of the time interval $I = [0, T]$ into half open subintervals $I_n := (t_{n-1}, t_n]$ of (time step) size $k_n := t_n - t_{n-1}$ with

$$0 = t_0 < \cdots < t_N = T.$$

Example 13.36 (Variational space-time of Navier-Stokes)**.** *For Navier-Stokes, we have*

$$\overline{A}(U)(\Psi) := \int_I A(U)(\Psi)\, dt$$

with

$$A(U)(\Psi) := (\partial_t v, \psi^v) + (v \cdot \nabla v, \psi^v) + (\sigma, \nabla \psi) + (\nabla \cdot v, \psi^p)$$

Moreover,

$$\overline{F}(\Psi) := \int_I (f, \psi^v)\, dt$$

The full variational form reads:

Formulation 13.37 (Space-time formulation NSE)**.** *Find $U := (v, p) \in X$ such that*

$$\overline{A}(U)(\Psi) + (v(0) - v^0, \psi^v(0)) = \overline{F}(\Psi) \quad \forall \Psi := (\psi^v, \psi^p) \in X$$

with

$$X := \{U = (v, p) | v \in L^2(I, H_0^1(\Omega)^n), \partial_t v \in L^2(I, L^2(\Omega)^n, p \in L^2(I, L^2(\Omega)/\mathbb{R})$$

Since X can be continuously embedded into $C(\bar{I}, L^2(\Omega)^n) \times L^2(I, L^2(\Omega)/\mathbb{R})$ the expression of the initial data $v(0)$ makes sense for functions $U \in X$.

Remark 13.38. *The pressure is only defined up to a constant when only Dirichlet data are considered. Working for instance with the do-nothing condition implicitly normalizes the pressure on the outflow boundary and no pressure filter is necessary.*

13.16.1 Classifying the semi-linear forms of PDE systems

When we work with PDE systems or coupled problems, several PDEs arise. Here, it is useful to classify a priori the semi-linear forms into different categories.

Definition 13.39 (Arranging the semi-linear form into groups)**.** *We formally define the following semi-linear forms and group them into four categories:*

$$\textit{Stationary terms:} \qquad A_k \quad \Rightarrow A_S \tag{304}$$

$$\textit{Time-dependent terms:} \qquad A_k \quad \Rightarrow A_T, A_E, A_I \tag{305}$$

where A_S are stationary terms, A_T terms with time derivatives, A_E terms that are treated explicitly in a time-stepping scheme, A_I terms that are treated implicitly in a time-stepping scheme (e.g., pressure in Navier-Stokes).

We explain the term A_T in a bit more detail in the following. Since A_T contains the continuous time derivative, i.e.,

$$A_T(U)(\Psi) = (\partial_t(\ldots), \Psi)$$

we first need to discretize via a backward difference quotient:

$$A_T(U)(\Psi) \approx A_T(U^{n,k}) = \left(\frac{1}{k_n}(\ldots), \Psi\right).$$

13.16.2 Nonlinear time derivatives

When the time derivative is fully nonlinear we first apply the product rule and then apply the difference quotient with θ-weighting of the nonlinear coefficient.

We explain this procedure with the help of an

Example 13.40. *Let u and v be solution variables of a nonlinear system. Define $J := J(u)$ and let us consider:*

$$\partial_t(Jv) = J\partial_t v + v\partial_t J$$

Applying the difference quotient and theta-approximation for the term in front of the time dervative yields

$$J^\theta\Big(\frac{v - v^{n-1}}{k}\Big) + v^\theta\Big(\frac{J - J^{n-1}}{k}\Big)$$

with

$$J^\theta = \theta J^n + (1-\theta)J^{n-1}, \quad v^\theta = \theta v^n + (1-\theta)v^{n-1}$$

$$\hat{A}_T(\hat{U}^{n,k})(\hat{\Psi}) = \left(J^\theta\Big(\frac{v - v^{n-1}}{k}\Big), \psi\right) + \left(v^\theta\Big(\frac{J - J^{n-1}}{k}\Big), \psi\right) \tag{306}$$

Remark 13.41 (Convenience of the splitting)**.** *The splitting has no advantage except a clean notation. In more detail, it allows for a very compact notation when applying One-Step-θ and Fractional-Step-θ schemes [29] to nonlinear coupled PDE systems and multiphysics problems in which some terms are stationary and others are nonstationary; when some terms are treated explicitly and other terms fully implicitly. For instance, for temporal discretization of the Navier-Stokes equations, the pressure should be treated fully implicitly, thus no terms as $(1-\theta)\hat{p}_f^{n-1}$ should appear. Separating all terms in the above way yields immediately the correct temporal scheme as used for instance in [136]. Using this notational idea was one key part that facilitated a lot the implementation for the further extension to FSI plus PFF. Of course, it makes no difference in the code itself (expect its readability) if such a splitting is used or not. However, the readability and sustainability of programming code is extremely important. The goal should be to have a code that can be immediately used again by yourself and extended after some time (let's say three years of non-usage). Or, a code that is usable immediately by others, also here the authors has made very positive experiences with the code in [137]. For general comments on code development and its treatment we also refer the reader to [90].*

13.16.3 One-Step-θ and Fractional-Step-θ schemes

Definition 13.42. *Let the previous time step solution $\hat{U}^{n-1}$ and the time step $k := k_n = t_n - t_{n-1}$ be given. Find $\hat{U}^n$ such that*

$$\hat{A}_T(\hat{U}^{n,k})(\hat{\Psi}) + \theta\hat{A}_E(\hat{U}^n)(\hat{\Psi}) \tag{307}$$

$$+\hat{A}_P(\hat{U}^n)(\hat{\Psi}) + \hat{A}_I(\hat{U}^n)(\hat{\Psi}) \;=\; -(1-\theta)\hat{A}_E(\hat{U}^{n-1})(\hat{\Psi}) \tag{308}$$

$$+\theta\widehat{F}^n(\hat{\Psi}) + (1-\theta)\widehat{F}^{n-1}(\hat{\Psi}), \tag{309}$$

where $\widehat{F}^n(\hat{\Psi}) = (\hat{\rho}_s\hat{f}_s^n, \hat{\psi}_s^v)_{\widehat{\Omega}_s}$ with $\hat{f}_s^n := \hat{f}_s(t_n)$. The concrete scheme depends on the choice of the parameter θ as defined in Definition 12.11.

Definition 13.43. *Let us choose* $\theta = 1 - \frac{\sqrt{2}}{2}, \theta' = 1 - 2\theta$, *and* $\alpha = \frac{1-2\theta}{1-\theta}, \beta = 1 - \alpha$. *The time step is split into three consecutive sub-time steps. Let* $\hat{U}^{n-1}$ *and the time step* $k := k_n = t_n - t_{n-1}$ *be given.*

Find $\hat{U}^n$ *such that*

$$\hat{A}_T(\hat{U}^{n-1+\theta,k})(\hat{\Psi}) + \alpha\theta\hat{A}_E(\hat{U}^{n-1+\theta})(\hat{\Psi}) \tag{310}$$

$$+\theta\hat{A}_P(\hat{U}^{n-1+\theta})(\hat{\Psi}) + \hat{A}_I(\hat{U}^{n-1+\theta})(\hat{\Psi}) = -\beta\theta\hat{A}_E(\hat{U}^{n-1})(\hat{\Psi}) + \theta\hat{F}^{n-1}(\hat{\Psi}), \tag{311}$$

$$\tag{312}$$

$$\hat{A}_T(\hat{U}^{n-\theta,k})(\hat{\Psi}) + \beta\theta'\hat{A}_E(\hat{U}^{n-\theta})(\hat{\Psi}) \tag{313}$$

$$+\theta'\hat{A}_P(\hat{U}^{n-\theta})(\hat{\Psi}) + \hat{A}_I(\hat{U}^{n-\theta})(\hat{\Psi}) = -\alpha\theta'\hat{A}_E(\hat{U}^{n-1+\theta})(\hat{\Psi}) + \theta'\hat{F}^{n-\theta}(\hat{\Psi}), \tag{314}$$

$$\tag{315}$$

$$\hat{A}_T(\hat{U}^{n,k})(\hat{\Psi}) + \alpha\theta\hat{A}_E(\hat{U}^{n})(\hat{\Psi}) \tag{316}$$

$$+\theta\hat{A}_P(\hat{U}^{n})(\hat{\Psi}) + \hat{A}_I(\hat{U}^{n})(\hat{\Psi}) = -\beta\theta\hat{A}_E(\hat{U}^{n-1})(\hat{\Psi}) + \theta\hat{F}^{n-\theta}(\hat{\Psi}). \tag{317}$$

13.17 Resulting time discretized problems

With the help of the previous considerations, we formulate a statement for the time-discretized equations:

Formulation 13.44. *Let the semi-linear form* $\hat{A}(\cdot)(\cdot)$ *be formulated in terms of the previous arrangement, such that*

$$\hat{A}(\hat{U})(\hat{\Psi}) := \hat{A}_T(\hat{U})(\hat{\Psi}) + \hat{A}_I(\hat{U})(\hat{\Psi}) + \hat{A}_E(\hat{U})(\hat{\Psi}) + \hat{A}_P(\hat{U})(\hat{\Psi}).$$

After time discretization, let the time derivatives are approximated with

$$\hat{A}_T(\hat{U})(\hat{\Psi}) \approx \hat{A}_T(\hat{U}^{n,k})(\hat{\Psi}),$$

such that the time-discretized semi-linear form reads

$$\hat{A}(\hat{U}^n)(\hat{\Psi}) := \hat{A}_T(\hat{U}^{n,k})(\hat{\Psi}) + \hat{A}_I(\hat{U}^n)(\hat{\Psi}) + \hat{A}_E(\hat{U}^n)(\hat{\Psi}) + \hat{A}_P(\hat{U}^n)(\hat{\Psi}).$$

Then, we aim to find $\hat{U}^n \in \widehat{X}_D^0$ *for all* $n = 1, 2, \ldots, N$ *such that*

$$\hat{A}(\hat{U}^n)(\hat{\Psi}) = \widehat{F}(\hat{\Psi}) \quad \forall\hat{\Psi} \in \widehat{X},$$

where this equation is treated with one specific time-stepping scheme as introduced previously.

13.18 An academic example of finite-difference-in-time, Galerkin-FEM-in-space-discretization and linearization in a Newton setting

We collect ingredients from most other sections to discuss a final PDE, which is time-dependent and nonlinear. This specific PDE itself has no use neither in theory nor in practice and is purely academic, but is inspired by the conservation of momentum of a nonstationary, nonlinear, fluid-structure interaction problem [136].

We are given the following example:

Problem 13.45. *Let* $\Omega = (0, 1)$ *and* $I = (0, T)$ *with the end time value* $T > 0$. *Let the following PDE be given: Find* $v : R \times I \to \mathbb{R}$ *and* $u : R \times I \to R$, *for allmost all times, such that*

$$\partial_t^2 u - J\nabla \cdot \sigma(v)F^{-T} = f, \quad \text{in } \Omega, \quad \text{plus bc. and initial cond.},$$

and where $J := J(u), F := F(u)$ *and* $\sigma(v) = (\nabla v + \nabla v^T)$.

13.18.0.1 Time discretization We aim to apply a One-Step-θ scheme applied to the mixed problem:

$$\begin{aligned}\partial_t v - J\nabla\cdot\sigma(v)F^{-T} &= f,\\ \partial_t u - v &= 0.\end{aligned}$$

One-Step-θ discretization with time step size k, and $\theta\in[0,1]$, leads to

$$\begin{aligned}\frac{v-v^{n-1}}{k} - \theta J\nabla\cdot\sigma(v)F^{-T} - (1-\theta)J^{n-1}\nabla\cdot\sigma(v^{n-1})(F^{-T})^{n-1} &= \theta f + (1-\theta)f^{n-1},\\ \frac{u-u^{n-1}}{k} - \theta v - (1-\theta)v^{n-1} &= 0.\end{aligned}$$

13.18.0.2 Spatial pre-discretization: weak form on the continuous level We multiply by the time step k, apply with test functions from suitable spaces V and W and obtain the weak formulations

$$\begin{aligned}&(v-v^{n-1},\varphi) + k\theta(J\sigma(v)F^{-T},\nabla\varphi) + k(1-\theta)(J^{n-1}\sigma(v^{n-1})(F^{-T})^{n-1},\varphi)\\ =&k\theta(f,\varphi) + k(1-\theta)(f^{n-1},\varphi)\quad\forall\varphi\in V,\\ &(u-u^{n-1},\psi) + k\theta(v,\psi) + k(1-\theta)(v^{n-1},\psi) = 0\quad\forall\psi\in W.\end{aligned}$$

Sorting terms on left and right hand sides:

$$\begin{aligned}&(v,\varphi) + k\theta(J\sigma(v)F^{-T},\nabla\varphi)\\ =&(v^{n-1},\varphi) - k(1-\theta)(J^{n-1}\sigma(v^{n-1})(F^{-T})^{n-1},\varphi)\\ &\quad + k\theta(f,\varphi) + k(1-\theta)(f^{n-1},\varphi)\quad\forall\varphi\in V,\\ &(u,\psi) + k\theta(v,\psi) = (u^{n-1},\psi) - k(1-\theta)(v^{n-1},\psi)\quad\forall\psi\in W.\end{aligned}$$

13.18.0.3 A single semi-linear form - first step towards Newton solver Now, we build a single semi-linear form $A(\cdot)(\cdot)$ and right hand side $F(\cdot)$. Let[12] $U := \{v,u\}\in V\times W$ and $\Psi := \{\varphi,\psi\}\in V\times W$: Find $U\in V\times W$ such that:

$$\begin{aligned}A(U)(\Psi) &= (v,\varphi) + k\theta(J\sigma(v)F^{-T},\nabla\varphi) + (u,\psi) + k\theta(v,\psi)\\ F(\Psi) &= (v^{n-1},\varphi) - k(1-\theta)(J^{n-1}\sigma(v^{n-1})(F^{-T})^{n-1},\varphi) + k\theta(f,\varphi)\\ &\quad + k(1-\theta)(f^{n-1},\varphi)(u^{n-1},\psi) - k(1-\theta)(v^{n-1},\psi)\end{aligned}$$

for all $\Psi\in V\times W$.

13.18.0.4 Evaluation of directional derivatives - second step for Newton solver Let $\delta U := \{\delta v,\delta u\}\in V\times W$. Then the directional derivative of $A(U)(\Psi)$ is given by:

$$\begin{aligned}A'(U)(\delta U,\Psi) = (\delta v,\varphi) &+ k\theta\Big(J'(\delta u)\sigma(v)F^{-T} + J\sigma'(\delta v)F^{-T} + J\sigma(v)(F^{-T})'(\delta u),\nabla\varphi\Big)\\ &+ (\delta u,\psi) + k\theta(\delta v,\psi),\end{aligned}$$

where we applied the chain rule for the term $J\sigma(v)F^{-T}$. Here, in the 'non-prime' terms in the nonlinear part; namely J, F^{-T} and $\sigma(v)$, the previous Newton solution is inserted. We have now all ingredients to perform the Newton step (284).

[12] Of course, the solution spaces for ansatz- and test functions might differ.

13.18.0.5 Spatial discretization in finite-dimensional spaces In the final step, we assume conforming finite-dimensional subspaces $V_h \subset V$ and $W_h \subset W$ with $V_h := \{\varphi_1, \dots, \varphi_N\}$ and $W_h := \{\psi_1, \dots, \psi_M\}$. Then, the update solution variables in each Newton step are given by:

$$\delta v_h := \sum_{j=1}^{N} v_j \varphi_j, \quad \text{and} \quad \delta u_h := \sum_{j=1}^{M} u_j \psi_j.$$

For the linearized semi-linear form $A'(U)(\delta U, \Psi)$, we have:

$$A'(U_h)(\delta U_h, \Psi_h)$$

and with this (M_{ij} representing the entries of the Jacobian):

$$\begin{aligned} M = (M)_{ij} := A'(U_h)(\Psi_j, \Psi_i) = &\sum_{j=1}^{N} v_j(\varphi_j, \varphi_i) \\ &+ k\theta\Big(\sum_{j=1}^{M} u_j J(\psi_j)\sigma(v)F^{-T} + J(\sum_{j=1}^{N} v_j \sigma'(\varphi_j))F^{-T} + J\sigma(v)(\sum_{j=1}^{M} u_j (F^{-T})'(\psi_j)), \nabla\varphi_i\Big) \\ &+ \sum_{j=1}^{M} u_j(\psi_j, \psi_i) + k\theta \sum_{j=1}^{N} v_j(\varphi_j, \psi_i) \end{aligned}$$

for all test functions running through $i = 1, \dots, N, N+1, \dots, M$.

13.18.0.6 Newton's method and the resulting linear system We recall:

$$\begin{aligned} A'(U_h^{n,j})(\delta U_h^n, \phi) &= -A(U_h^{n,j})(\phi) + F(\phi), \\ U_h^{n,j+1} &= U_h^{n,j} + \lambda \delta U_h^n. \end{aligned}$$

The linear equation system reads in matrix form:

$$M\delta U = B \tag{318}$$

where M has been defined before, B is the discretization of the residual:

$$B \sim A(u_h^{n,j})(\phi) + F(\phi)$$

and the solution vector δU is given by

$$\delta U = (\delta v_1, \dots, \delta v_N, \delta u_1, \dots, \delta u_M)^T.$$

Since we dealt originally with two PDEs (now somewhat hidden in the semilinear form), it is often desirable to write (318) in block form:

$$\begin{pmatrix} M_{vv} & M_{vu} \\ M_{uv} & M_{uu} \end{pmatrix} \begin{pmatrix} \delta v \\ \delta u \end{pmatrix} = \begin{pmatrix} B_v \\ B_u \end{pmatrix}$$

where B_v and B_u are the residual parts corresponding to the first and second PDEs, respectively. Since in general such matrix systems are solved with iterative solvers, the block form allows a better view on the structure and construction of preconditioners. For instance, starting again from (318), a preconditioner is a matrix P^{-1} such that

$$P^{-1}M\delta U = P^{-1}B$$

so that the condition number of $P^{-1}M$ is moderate. Obviously, the ideal preconditioner would be the inverse of A: $P^{-1} = A^{-1}$. In practice one tries to build P^{-1} in such a way that

$$P^{-1}M = \begin{pmatrix} I & * \\ 0 & I \end{pmatrix}$$

and where P^{-1} is a lower triangular block matrix:

$$P^{-1} = \begin{pmatrix} P_1^{-1} & 0 \\ P_3^{-1} & P_4^{-1} \end{pmatrix}$$

The procedure is as follows (see lectures for linear algebra in which the inverse is explicitely constructed):

$$\begin{aligned} M &= \begin{pmatrix} M_{vv} & M_{vu} \\ M_{uv} & M_{uu} \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix} \\ &= \begin{pmatrix} I & M_{vv}^{-1} M_{vu} \\ M_{uv} & M_{uu} \end{pmatrix} \begin{pmatrix} M_{vv}^{-1} & 0 \\ 0 & I \end{pmatrix} \\ &= \begin{pmatrix} I & M_{vv}^{-1} M_{vu} \\ 0 & \underbrace{M_{uu} - M_{uv} M_{vv}^{-1} M_{vu}}_{=S} \end{pmatrix} \begin{pmatrix} M_{vv}^{-1} & 0 \\ -M_{uv} M_{vv}^{-1} & I \end{pmatrix} \\ &= \begin{pmatrix} I & M_{vv}^{-1} M_{vu} \\ 0 & I \end{pmatrix} \underbrace{\begin{pmatrix} M_{vv}^{-1} & 0 \\ -S^{-1} M_{uv} M_{vv}^{-1} & S^{-1} \end{pmatrix}}_{=P^{-1}} \end{aligned}$$

where $S = M_{uu} - M_{uv} M_{vv}^{-1} M_{vu}$ is the so-called **Schur complement**. The matrix P^{-1} is used as (exact) preconditioner for M. Indeed we double-check:

$$\begin{pmatrix} M_{vv}^{-1} & 0 \\ -S^{-1} M_{uv} M_{vv}^{-1} & S^{-1} \end{pmatrix} \begin{pmatrix} M_{vv} & M_{vu} \\ M_{uv} & M_{uu} \end{pmatrix} = \begin{pmatrix} I & M_{vv}^{-1} M_{vu} \\ 0 & I \end{pmatrix}$$

Tacitely we assumed in the entire procedure that S and M_{vv} are invertible.

Using P^{-1} in a Krylov method, we only have to perform matrix-vector multiplications such as

$$\begin{pmatrix} X_{new} \\ Y_{new} \end{pmatrix} = \begin{pmatrix} P_1^{-1} & 0 \\ P_3^{-1} & P_4^{-1} \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix}$$

Now we obtain:

$$X_{new} = P_1^{-1} X \tag{319}$$

$$Y_{new} = P_3^{-1} X + P_4^{-1} Y \tag{320}$$

Remark 13.46. *Be careful, we deal with two iterative procedures in this example: Newton's method to compute iteratively the nonlinear solution. Inside Newton's method, we solve the linear equations systems with a Krylov space method, which is also an iterative method.*

13.19 Navier-Stokes - FEM discretization

In a similar way to Section 11.5, we briefly present the NSE FEM discretization. As we previously discusssed, rather than solving directly for v_h and p_h we solve now for the (Newton) updates δv_h and δp_h - later more.

The problem reads:

$$\text{Find } U_h \in X_h \text{ such that: } A(U_h)(\Psi_h) = F(\Psi_h) \quad \forall \Phi \in X_h,$$

where

$$\begin{aligned} A(U_h)(\Psi_h) &= (v_h \cdot \nabla v_h, \psi_h^v) + \frac{1}{Re}(\nabla v_h, \nabla \psi_h^v) - (p_h, \nabla \cdot \psi_h^v) + (\nabla \cdot v_h, \psi_h^p), \\ F(\Psi_h) &= (f_f, \psi_h^v). \end{aligned}$$

Here, we added the dimensionless Reynolds number Re in order to be aware how the equations change their type if Re is either small or large.

Remark 13.47 (Notations for bilinear and semilinear forms)**.** *If the PDE is linear, we use the notation for a bilinear form: $A(U_h, \Psi_h)$. If the PDE is nonlinear, this is indicated in the notation by using $A(U_h)(\Psi_h)$.*

In the following, we apply Newton's method. Given an initial guess $U_h^0 := \{v_h^0, p_h^0\}$, we must solve the problem:

$$\text{Find } \delta U_h := \{\delta v_h, \delta p_h\} \in X_h \text{ such that: } A'(U_h^l)(\delta U_h, \Psi_h) = -A(U_h^l)(\Psi_h) + F(\Psi_h), \quad U_h^{l+1} = U_h^l + \delta U_h.$$

Here,

$$\begin{aligned} A'(U_h^l)(\delta U_h, \Psi_h) &= (\delta v_h \cdot \nabla v_h^l + v_h^l \cdot \nabla \delta v_h, \psi_h^v) + \frac{1}{Re}(\nabla \delta v_h, \nabla \psi_h^v) - (\delta p_h, \nabla \cdot \psi_h^v) + (\nabla \cdot \delta v_h, \psi_h^p), \\ F(\Psi_h) &= (f_f, \psi_h^v), \\ A(U_h^l)(\Psi_h) &= (v_h^l \cdot \nabla v_h^l, \psi_h^v) + \frac{1}{Re}(\nabla v_h^l, \nabla \psi_h^v) - (p_h^l, \nabla \cdot \psi_h^v) + (\nabla \cdot v_h^l, \psi_h^p). \end{aligned}$$

As explained, we solve now for the updates and their respresentation with the help of the shape functions:

$$\delta v_h = \sum_{j=1}^{N_V} \delta v_j \psi_h^{v,j}, \quad \delta p_h = \sum_{j=1}^{N_P} \delta p_j \psi_h^{p,j}$$

With that the discrete, linearized convection operator reads:

$$L := (\psi_h^{v,j} \cdot \nabla v_h^l + v_h^l \cdot \nabla \psi_h^{v,j}, \psi_h^{v,i})_{ij=1}^{N_V, N_V}.$$

The block structure reads:

$$\begin{pmatrix} L + \frac{1}{Re}A & B \\ -B^T & 0 \end{pmatrix} \begin{pmatrix} \delta v \\ \delta p \end{pmatrix} = \begin{pmatrix} f - [L^l + \frac{1}{Re}A^l + B] \\ 0 - [-B^T]^l \end{pmatrix},$$

where we have added on the right hand side the vectors of Newton's residual.

Remark 13.48. *A numerical test is presented in Section 9.24.*

13.20 Newton solver performance for Poiseuille flow in a channel

We take DOpElib www.dopelib.net and go to dopelib/Examples/PDE/StatPDE/Example1. The original code takes the (linear) Stokes equations. Since we always use a nonlinear solver, we should obtain convergence in one single Newton step.

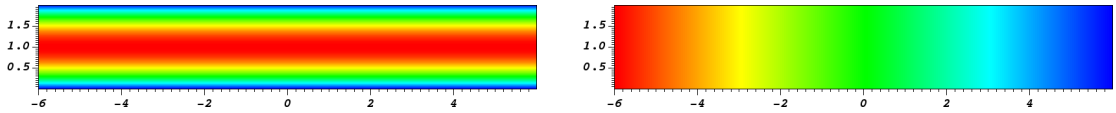


Figure 65: Poiseuille flow in a channel: x-velocity and pressure p.

13.20.1 Stokes

First the absolute residual value is set to a relative value. Then, the problem converges in one Newton step.

```
Newton step: 0  Residual (abs.):   8.83333e-01
Newton step: 0  Residual (rel.):   1.00000e+00
Newton step: 1  Residual (rel.):   2.84364e-15
```

13.20.2 Navier-Stokes with $\nu_f = 1$

With our previous formulation, we have here $\nu_f = \frac{1}{Re}$. We obtain:

```
Newton step: 0  Residual (abs.):   8.99844e-01
Newton step: 0  Residual (rel.):   1.00000e+00
Newton step: 1  Residual (rel.):   4.24110e-02
Newton step: 2  Residual (rel.):   5.92363e-07
Newton step: 3  Residual (rel.):   1.30338e-10
Newton step: 4  Residual (rel.):   1.72731e-14
```

This is quadratic convergence and considered a 'optimal' Newton convergence for such problems.

13.20.3 Navier-Stokes with $\nu_f = 1e-2$

We shift more to convection-dominated flow and obtain now:

```
Newton step: 0  Residual (abs.):   1.03895e-01
Newton step: 0  Residual (rel.):   1.00000e+00
Newton step: 1  Residual (rel.):   8.47208e-01  LineSearch {3}
Newton step: 2  Residual (rel.):   1.78723e-01  LineSearch {0}
Newton step: 3  Residual (rel.):   2.95891e-02  LineSearch {0}
Newton step: 4  Residual (rel.):   7.34315e-04  LineSearch {0}
Newton step: 5  Residual (rel.):   2.07151e-06  LineSearch {0}
Newton step: 6  Residual (rel.):   1.56055e-08  LineSearch {0}
Newton step: 7  Residual (rel.):   9.93743e-11  LineSearch {0}
```

In particular at the beginning, the Newton solver has now more problems and even needs some backtracking line search steps in order to converge. In the steps 4 - 7 the convergence is fast, but not quadratic anymore.

13.21 Chapter summary and outlook

In this section we provided an introduction to nonlinear problems. Most importantly we introduced numerical solver techniques such as fixed-point schemes and Newton methods. These are illustrated with several examples; finally we addressed descriptions for solving nonlinear, nonstationary, multiphysics problems. In the next chapter, we focus on situations in which two (or more) PDEs couple and interact with each other.

14 Coupled problems

In this chapter, we address these topics and explain how two or more PDEs can be coupled on a variational level. In practice, the coupling of different PDEs or PDEs with different coefficients in different subdomains is a timely research topic (the keyword is **multiphysics** problems[13]).

Before we start, we refer in particular as well to:

- Section 4.1.7 in which an example of a linear coupled problem is given;
- Section 5.5 in which an explanation-only-in-words of coupled PDEs is given;

As in the previous chapter, we also refer the reader to my book [141].

14.1 An interface-coupled problem in 1D

In this section, we explain the basic features of an interface-coupled problem in 1D.

Formulation 14.1. *Let $\Omega = (a,b)$ be an open domain and $k := k(x) \in \mathbb{R}$ a material coefficient, which is specified as*

$$k(x) = \begin{cases} k_1, \text{ for } x \in \Omega_1 = (a,c), \\ k_2, \text{ for } x \in \Omega_2 = (c,b) \end{cases}$$

for $a < c < b$. The solution is split into two parts:

$$u(x) = \begin{cases} u_1(x), \text{ for } x \in \Omega_1, \\ u_2(x), \text{ for } x \in \Omega_2, \end{cases}$$

which is obtained by solving

$$\begin{cases} -k_1 u_1'' = f \text{ in } \Omega_1, \\ -k_2 u_2'' = f \text{ in } \Omega_2, \\ u_1(a) = u_2(b) = 0, \\ u_1(c) = u_2(c), \\ k_1 u_1'(c) = k_2 u_2'(c). \end{cases} \tag{321}$$

The third line in (321) *is well-known to us and denotes homogeneous Dirichlet conditions on the outer boundaries. Afterwards, we have two new conditions that are so-called **interface** or **coupling** conditions and which are very characteristic to coupled problems.*

Remark 14.2. *One of the most famous examples of a coupled (multiphysics) problem is fluid-structure interaction [112, 138] in which the Navier-Stokes equations (fluid flow) are coupled to elasticity (solid mechanics).*

In the following, we now derive a variational formulation. Again the first step is to formulate a function space. To this end, we define:

$$V = \{u \in \mathcal{C}^0(\bar{\Omega}) : u|_{\Omega_1} \in \mathcal{C}^1(\Omega_1), u|_{\Omega_2} \in \mathcal{C}^1(\Omega_2) \text{ and } u(a) = u(b) = 0\}.$$

Remark 14.3 (on the Dirichlet coupling conditions)**.** *Since we seek $u \in \mathcal{C}^0(\bar{\Omega})$ the first coupling condition $u_1(c) = u_2(c)$ is implicitly contained. In general, however, it holds that same rule as before: Dirichlet conditions need to be set explicitely in the function space and Neumann conditions, i.e., the second coupling condition, appears naturally in the equations through integration by parts.*

It holds:

[13] The word multiphysics is self-explaining: here multiple physical phenomena interact. Examples are interaction of solids with temperature, interaction of solids with fluids, interaction of fluids with chemical interactions, interaction of solids with electromagnetic waves.

Proposition 14.4 (Variational formulation of the coupled problem)**.** *Find $u \in V$ such that*

$$a(u,\phi) = l(\phi) \quad \forall \phi \in V$$

with

$$a(u,\phi) = \int_a^b k(x)u'(x)\phi'(x)\,dx, \tag{322}$$

$$l(\phi) = \int_a^b f(x)\phi(x)\,dx. \tag{323}$$

Proof. As in the previous sections, we first multiply with a test function from V and integrate:

$$-\int_a^c k_1 u_1''(x)\phi_1\,dx - \int_c^b k_2 u_2''(x)\phi_1\,dx = \int_a^b f(x)\phi\,dx. \tag{324}$$

We perform integration by parts:

$$\int_a^c k_1 u_1'(x)\phi_1'(x)\,dx - k_1\Big[u_1'(c)\phi_1(c) - u_1'(a)\phi_1(a)\Big] \tag{325}$$

$$+\int_c^b k_2 u_2'(x)\phi_2'(x)\,dx - k_2\Big[u_2'(b)\phi_2(b) - u_2'(c)\phi_2(c)\Big] = \int_a^b f(x)\phi(x)\,dx \tag{326}$$

using the continuity (first coupling condition) of v on the interface c, we obtain:

$$\int_a^b k(x)u'(x)\phi'(x)\,dx - \underbrace{\Big[k_1u_1'(c) - k_2u_2'(c)\Big]}_{=0}\phi(c) + k_1u_1'(a)\underbrace{\phi_1(a)}_{=0} - k_2u_2'(b)\underbrace{\phi_2(b)}_{=0} = \int_a^b f(x)\phi(x)\,dx. \tag{327}$$

On the outer boundaries, the boundary terms will vanish because the test functions are zero. On the interface, the terms will vanish as well since we can employ the second coupling condition. Consequently, we obtain (322) with $u \in V$. □

We next show the equivalence of solutions between the strong form and the variational forms:

Proposition 14.5. *Let $u \in C^2$ and $u|_{\Omega_i} \in \mathcal{C}^2(\overline{\Omega}_1), i = 1,2$. Then, u is a solution of the strong form* (321) *if and only if u is a solution of the variational problem.*

Proof. As usually, we first show $(D) \to (V)$. Let u be a solution of (321). Moreover, u has sufficient regularity such that the previous calculations (multiplication with a test function, integration, integration by parts) hold true and $u \in V$. Therefore, u is a solution of (V).

We now discuss $(V) \to (D)$ and consider $u \in V$ to be solution of (V) plus the assumption that u is twice differentiable in each subdomain Ω_i. Consequently, we can integrate by parts backwards. Let $\phi \in V$. Then:

$$-\int_a^c k_1 u_1''(x)\phi_1(x)\,dx + k_1\Big[u_1'(c)\phi_1(c) - u_1'(a)\phi_1(a)\Big] \tag{328}$$

$$-\int_c^b k_2 u_2''(x)\phi_2(x)\,dx + k_2\Big[u_2'(b)\phi_2(b) - u_2'(c)\phi_2(c)\Big] = \int_a^b f(x)\phi(x)\,dx \tag{329}$$

On the boundary $\bar{\Omega}$ we have zero test functions V, and obtain

$$-\int_a^c k_1 u_1''(x)\phi_1(x)\,dx + k_1u_1'(c)\phi_1(c) - \int_c^b k_2 u_2''(x)\phi_2(x)\,dx - k_2u_2'(c)\phi_2(c) = \int_a^b f(x)\phi(x)\,dx. \tag{330}$$

In the following we now discuss term by term on each interval:

- We choose a subspace of V such that ϕ_2 is zero and ϕ_1 has a compact support on Ω_1 (yielding specifically that $\phi_1(a) = \phi_1(c) = 0$). Such a subspace is indeed a subspace of V since it is continuous on c and $\phi(a) = \phi(b) = 0$. Consequently, we obtain

$$-k_1 u_1''(x) = f(x) \quad a < x < c.$$

  The other way around holds true as well from which we get

$$-k_2 u_2''(x) = f(x) \quad c < x < b.$$

  Consequently, we have recovered the two differential equations inside Ω_1 and Ω_2.

- We now discuss the second interface conditions. In particular, the previous findings from (330) and eliminating the boundary terms give us for $\phi \in V$:

$$k_1 u_1'(c)\phi_1(c) - k_2 u_2'(c)\phi_2(c) = 0.$$

  The continuity of ϕ at c becomes now essential such that we can write:

$$\Big[k_1 u_1'(c) - k_2 u_2'(c)\Big]\phi(c) = 0, \quad \forall \phi \in V.$$

  We now construct $\phi \in V$ such that $\phi(c) = 1$ (take for instance piece-wise linear functions on each subdomain). This yields the transmission conditions:

$$k_1 u_1'(c) - k_2 u_2'(c) = 0.$$

- Finally, we also get the missing conditions on the outer boundary and the first interface conditions because $u \in V$:

$$u(a) = u(b) = 0$$

  for $u \in V$. Finally, since $u \in V$, i.e., $u \in C^0$ yields

$$u_1(c) = u_2(c).$$

In summary, u is a solution of (321), which shows that the variational formulation yields the classical solution. □

Exercise 11. *Exercise 2.8 in [82].*

14.2 Volume coupling of two PDEs

Let $\Omega \subset \mathbb{R}^n$ and $\partial\Omega$ be a sufficiently smooth boundary. Let V_1 and V_2 be Hilbert or Sobolev spaces that are appropriate to each single PDE. For simplicity, we work with homogeneous Dirichlet conditions on $\partial\Omega$. In volume-coupling both equations are defined in the same domain Ω (this is important here!).

In a formal way, we write for two variational forms:

$$\text{Find } u_1 \in V_1: \quad A_1(\{u_1, u_2\})(\varphi_1) = F_1(\varphi_1) \quad \forall \varphi_1 \in V_1, \tag{331}$$

$$\text{Find } u_2 \in V_2: \quad A_2(\{u_1, u_2\})(\varphi_2) = F_2(\varphi_2) \quad \forall \varphi_2 \in V_2, \tag{332}$$

where A_1 and A_2 are semi-linear forms representing the partial differential operators. Specifically, the solution variable of the other problem may appear in each equation.

14.3 Partitioned versus monolithic coupling

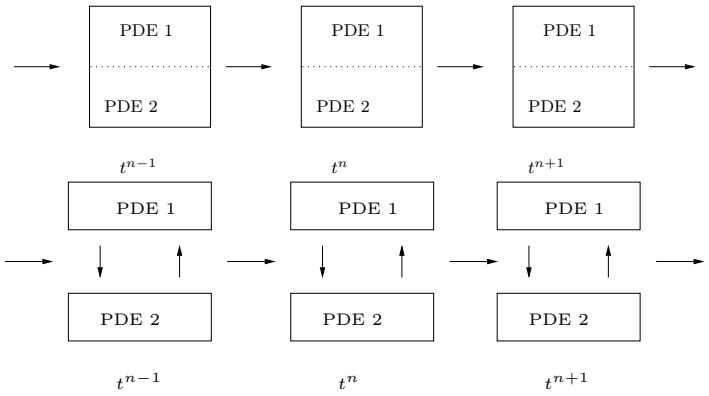


Figure 66: Monolithic and paritioned coupling schemes. In monolithic coupling, the coupled system is solved all-at-once whereas in the paritioned scheme subiterations are required to enforce the force balance on the interface.

14.4 Iterative coupling (known as partitioned or staggered) schemes

For coupled problems, very often, partitioned (or also so-called staggered) schemes are employed. The idea is to iterate between the different equations per time step.

Find $U = (u, v) \in V \times W$ such that:

$$A_u(U)(\psi^u) = F(\psi^u), \tag{333}$$
$$A_v(U)(\psi^v) = F(\psi^v). \tag{334}$$

The idea is to iterate between both PDEs:

Algorithm 14.6. *Given the initial iteration iteration values $(u^0, v^0) \in V \times W$. For $k = 1, 2, 3, \ldots$ iterate:*

$$\textit{Given } v^{k-1}\textit{, find } u^k\textit{:} \quad A_u(u^k, v^{k-1})(\psi^u) = F(\psi^u), \tag{335}$$
$$\textit{Given } u^k\textit{, find } v^k\textit{:} \quad A_v(u^k, v^k)(\psi^v) = F(\psi^v). \tag{336}$$

Check the stopping criterion:

$$\max(\|u^k - u^{k-1}\|, \|v^k - v^{k-1}\|) < TOL.$$

If the stopping criterion is fulfilled, stop. If not, increment $k \to k + 1$.

14.5 Variational-monolithic coupling

One possibility is a variational-monolithic coupling (in short often monolithic) in which we first define the space

$$X = V_1 \times V_2.$$

This allows us to define a compact semi-linear form with both solution variables. To this end, let $U := (u_1, u_2) \in X$. This allows us to sum-up both equations such that

$$\text{Find } U \in X: \quad A(U)(\Psi) = F(\Psi) \quad \forall \Psi \in X,$$

where $\Psi = (\varphi_1, \varphi_2) \in X$ and

$$A(U)(\Psi) := A_1(\{u_1, u_2\})(\varphi_1) + A_2(\{u_1, u_2\})(\varphi_2).$$

If the problem is linear, $A(U)(\Psi)$ can be treated with a linear solver (Direct, CG, GMRES, MG, ...) with appropriate preconditioners. In fact, the development of a good preconditioner for coupled problems is very often the most challenging part.

If $A(U)(\Psi)$ is nonlinear (which is in most coupled problems the case), then $A(U)(\Psi)$ can be solved with the techniques explained in Chapter 13.

Remark 14.7. *This concept can be easily extended to n PDEs rather than only* 2.

14.6 Examples

14.6.1 Two stationary elliptic PDEs

We couple two elliptic PDEs. The first describes some displacements while the second is a heat transfer problem.

Find $u : \Omega \to \mathbb{R}$ such that

$$\begin{aligned} -\nabla \cdot (T \nabla u) &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned}$$

and find $T : \Omega \to \mathbb{R}$ such that

$$\begin{aligned} -\nabla \cdot (\alpha \nabla T) &= g(u) \quad \text{in } \Omega, \\ T &= 0 \quad \text{on } \partial\Omega. \end{aligned}$$

The function spaces read:

$$\begin{aligned} V_1 &:= H_0^1(\Omega) \\ V_2 &:= H_0^1(\Omega). \end{aligned}$$

The product space reads:

$$X := V_1 \times V_2.$$

This allows us to define $U := (u, T) \in X$ and the following compact semi-linear form:

$$\begin{aligned} A(U)(\Psi) := &(\partial_t u, \psi_1) + (T \nabla u, \nabla \psi_1) \\ &+ (\partial_t T, \psi_2) + (\alpha \nabla T, \nabla \psi_2) - (g(u), \psi_2) \end{aligned}$$

and the right hand side:

$$F(\Psi) := (f, \psi_1).$$

Then, we obtain:

Formulation 14.8. *Find $U \in X$ such that*

$$A(U)(\Psi) = F(\Psi) \quad \forall \Psi \in X.$$

14.6.2 Two time-dependent parabolic PDEs

We now extend the previous equations, to a time-dependent situation and couple two parabolic equations. Find $u : \Omega \times I \to \mathbb{R}$ such that

$$\begin{aligned} \partial_t u - \nabla \cdot (T \nabla u) &= f \quad \text{in } \Omega \times I, \\ u &= 0 \quad \text{on } \partial\Omega \times I, \\ u(0) &= u_0 \quad \text{in } \partial\Omega \times \{0\}, \end{aligned}$$

and find $T : \Omega \times I \to \mathbb{R}$ such that

$$\begin{aligned} \partial_t T - \nabla \cdot (\alpha \nabla T) &= g(u) \quad \text{in } \Omega \times I, \\ T &= 0 \quad \text{on } \partial\Omega \times I, \\ T(0) &= T_0 \quad \text{in } \partial\Omega \times \{0\}, \end{aligned}$$

The Bochner function spaces read:

$$V_1 := L^2(I, H_0^1(\Omega))$$
$$V_2 := L^2(I, H_0^1(\Omega)).$$

The product space reads:

$$X := V_1 \times V_2.$$

This allows us to define $U := (u, T) \in X$ and the following compact semi-linear form:

$$A(U)(\Psi) := \int_I (\partial_t u, \psi_1) + (T\nabla u, \nabla\psi_1)\, dt + (u(0) - u_0, \psi_1(0))$$
$$+ \int_I (\partial_t T, \psi_2) + (\alpha\nabla T, \nabla\psi_2) - (g(u), \psi_2)\, dt + (T(0) - T_0, \psi_2(0))$$

and the right hand side:

$$F(\Psi) := \int_I (f, \psi_1)\, dt.$$

Then, we obtain:

Formulation 14.9. *Find $U \in X$ such that*

$$A(U)(\Psi) = F(\Psi) \quad \forall \Psi \in X.$$

14.6.3 The Biot problem

The standard Biot problem in Section 4.1.7 is an example of a volume-coupled problem.

14.7 Interface coupling

14.7.1 Preliminaries: Interface-Tracking and Interface-Capturing

Interface coupling is more challenging than volume-coupling since the PDEs live in different subdomains. Two basic methods can be distinguished:

- Interface-Tracking
- Interface-Capturing

In methods, where the domain is decomposed into elements or cells (finite volumes, finite elements or isogeometric analysis), using interface-tracking aligns mesh edges with the interface. These are so-called **fitted methods**. For moving interfaces, the mesh elements need to be moved as well; see Figure 67. However, mesh elements may be deformed too much such that the approach fails if not taken care of (expensive) re-meshing in a proper way.

In interface-capturing methods (unfitted methods), the domain and consequently the single elements stay fixed; see Figure 67 (left). Here, the interface can move freely through the domain. Mesh degeneration is not a problem, but capturing the interface is difficult. In this approach a further classification can be made:

- Lower-dimensional approaches
- Diffusive techniques.

The first method comprises extended/generalized finite elements, cut-cell methods, finite cell methods, and locally modified finite elements. Diffusive methods are the famous level-set method or phase-field methods.

Short summary Finally, using interface-tracking or interface-capturing approaches are a compromise between computational and implementation efforts and the accuracy of the desired interface approximation. While in general interface-capturing are easier to implement and can deal in an easier way with moving and evolving interfaces, the accuracy for the same number of degrees of freedom is lower than comparable interface-tracking approaches. The latter are, however, more challenging when interfaces are moving, propagating - in particular in 3D. To this end, as said just before: it is a compromise as many things in life.

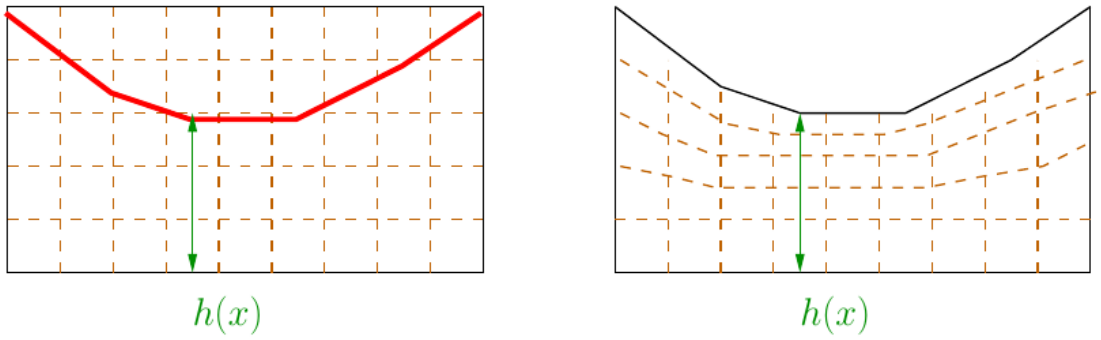


Figure 67: Left: the mesh is fixed and the interface must be captured. Right: interface-tracking in which the interface is located on mesh edges.

14.7.2 Interface coupling of two PDEs

We now address interface coupling in more detail. Here, both (or multiple) PDEs live in different subdomains. Let $\Omega_1 \subset \mathbb{R}^n$ and $\Omega_2 \subset \mathbb{R}^n$ and $\Omega := \bar{\Omega}_1 \cup \bar{\Omega}_2$. The interface is described by

$$\Gamma := \bar{\Omega}_1 \cap \bar{\Omega}_2.$$

Let both the outer boundary $\partial\Omega$ and the interface Γ both be sufficiently smooth.

Let $V_1(\Omega_1)$ and $V_2(\Omega_2)$ be Hilbert or Sobolev spaces that are appropriate to each single PDE. For simplicity, we work with homogeneous Dirichlet conditions.

Let $u_1 : \Omega_1 \to \mathbb{R}$ and $u_2 : \Omega_2 \to \mathbb{R}$. For second-order equations (for instance two Poisson problems) we need to impose two interface conditions on Γ:

$$\begin{aligned} u_1 &= u_2 \\ \partial_{n_1} u_1 &= \partial_{n_2} u_2, \end{aligned}$$

with the relation of the normal vectors $n_1 = -n_2$. In the first is known as kinematic condition and the second is the continuity of normal forces. The first condition is a Dirichlet-like condition, which needs to be imposed in the function space. The second one is a Neumann condition that will vanish in the weak formulation when a variational-monolithic coupling algorithm is used.

In a formal way, we write for two variational forms:

$$\begin{aligned} &\text{Find } u_1 \in V_1 : \quad A_1(\{u_1, u_2\})(\varphi_1) = F_1(\varphi_1) \quad \forall \varphi_1 \in V_1, \\ &\text{Find } u_2 \in V_2 : \quad A_2(\{u_1, u_2\})(\varphi_2) = F_2(\varphi_2) \quad \forall \varphi_2 \in V_2, \end{aligned}$$

14.7.3 Variational-monolithic coupling

One possibility is a variational-monolithic coupling (in short often monolithic) in which, we first define the space

$$X = \{v \in H_0^1(\Omega) |\ v_1 = v_2 \text{ on } \Gamma\}$$

Here, the function v is defined over the entire domain Ω. Moreover, the kinematic coupling condition is built into X.

This allows us to define a compact semi-linear form with both solution variables. To this end, let $U := (u_1, u_2) \in X$. This allows us to sum-up both equations such that

Proposition 14.10.

$$\textit{Find } U \in X : \quad A(U)(\Psi) = F(\Psi) \quad \forall \Psi \in X,$$

where $\Psi = (\varphi_1, \varphi_2) \in X$ *and*

$$A(U)(\Psi) := A_1(\{u_1, u_2\})(\varphi_1) + A_2(\{u_1, u_2\})(\varphi_2).$$

Proof. It holds after partial integration of the strong forms:

$$A_1(\{u_1, u_2\})(\varphi_1) + \int_\Gamma \partial_n u_1 \varphi_1 \, ds$$

and

$$A_2(\{u_1, u_2\})(\varphi_2) + \int_\Gamma \partial_n u_2 \varphi_2 \, ds.$$

We sum-up:

$$\begin{aligned} &A_1(\{u_1, u_2\})(\varphi_1) + \int_\Gamma \partial_n u_1 \varphi_1 \, ds + A_2(\{u_1, u_2\})(\varphi_2) + \int_\Gamma \partial_n u_2 \varphi_2 \, ds \\ &= A_1(\{u_1, u_2\})(\varphi_1) + A_2(\{u_1, u_2\})(\varphi_2) + \int_\Gamma (\partial_n u_1 \varphi_1 + \partial_n u_2 \varphi_2) \, ds. \end{aligned}$$

We argue that $\Psi = (\varphi_1, \varphi_2) \in X$. In particular, therein $\varphi_1 = \varphi_2$ on Γ. Also we use the fact that $n_1 = -n_2$. Then:

$$\int_\Gamma (\partial_{n_1} u_1 \varphi_1 + \partial_{n_2} u_2 \varphi_1) \, , ds = \int_\Gamma (\partial_{n_1} u_1 + \partial_{n_1} u_2) \varphi_1 \, ds$$

Our second coupling condition $\partial_{n_1} u_1 + \partial_{n_1} u_2 = 0$ comes into play now and therefore the integral on Γ vanishes:

$$\int_\Gamma (\partial_{n_1} u_1 + \partial_{n_1} u_2) \varphi_1 \, ds = 0.$$

Therefore, the sum of both semi-linear forms does not contain any terms on the interface Γ and consequently, we have

$$A(U)(\Psi) := A_1(\{u_1, u_2\})(\varphi_1) + A_2(\{u_1, u_2\})(\varphi_2).$$

□

14.8 Examples

14.8.1 Two stationary elliptic PDEs

We study two elliptic PDEs in different domains.

Find $u_1 : \Omega_1 \to \mathbb{R}$ such that

$$\begin{aligned} -\nabla \cdot (u_2 \nabla u_1) &= f \quad \text{in } \Omega_1, \\ u_1 &= 0 \quad \text{on } \partial\Omega_1, \end{aligned}$$

and find $u_2 : \Omega_2 \to \mathbb{R}$ such that

$$\begin{aligned} -\nabla \cdot (\alpha \nabla u_2) &= g(u_1) \quad \text{in } \Omega_2, \\ u_2 &= 0 \quad \text{on } \partial\Omega_2 \end{aligned}$$

where $\alpha > 0$ and some diffusion coefficient.

We impose the following coupling conditions on Γ:

$$\begin{aligned} u_1 &= u_2 \\ u_2 \nabla u_1 \cdot n_1 &= \alpha \nabla u_2 \cdot n_2. \end{aligned}$$

Using monolithic coupling the joint function space read:

$$X = \{v \in H_0^1(\Omega) |\ v_1 = v_2 \text{ on } \Gamma\}$$

This allows us to define $U := (u, T) \in X$ and the following compact semi-linear form:

$$\begin{aligned} A(U)(\Psi) := &(\partial_t u, \psi_1) + (T \nabla u, \nabla \psi_1) \\ &+ (\partial_t T, \psi_2) + (\alpha \nabla T, \nabla \psi_2) - (g(u), \psi_2) \end{aligned}$$

and the right hand side:

$$F(\Psi) := (f, \psi_1).$$

Then, we obtain:

Formulation 14.11. *Find $U \in X$ such that*

$$A(U)(\Psi) = F(\Psi) \quad \forall \Psi \in X.$$

This formulation is identical to volume-coupling, but the fact is hidden that u_1 and u_2 live on different subdomains.

14.8.2 The augmented Biot-Lamé-Navier problem

The Biot problem coupled to elasticity in Section 4.1.7 is an example in which a volume-coupled problem is coupled via interface-coupled to another problem. Here, the interface conditions must be very carefully described and implemented.

14.9 Variational-monolithic fluid-structure interaction

In this section, we present an example of a nonlinear, nonstationary interface-coupled problem: fluid-structure interaction. FSI can be written in terms of semi-linear forms (see Chapter 14) that allow later for compact notation:

Problem 14.12 (ALE$_{fx}$ FSI with harmonic and linear-elastic mesh motion)**.** *Find $\hat{U} := \{\hat{v}_f, \hat{v}_s, \hat{u}_f, \hat{u}_s, \hat{p}_f, \hat{p}_s\} \in \widehat{X}_D^0 := \{\hat{v}_f^D + \hat{V}_{f,\hat{v}}^0\} \times \hat{L}_s \times \{\hat{u}_f^D + \hat{V}_{f,\hat{u}}^0\} \times \{\hat{u}_s^D + \hat{V}_s^0\} \times \hat{L}_f^0 \times \hat{L}_s^0$, such that $\hat{v}_f(0) = \hat{v}_f^0$, $\hat{v}_s(0) = \hat{v}_s^0$, $\hat{u}_f(0) = \hat{u}_f^0$, and $\hat{u}_s(0) = \hat{u}_s^0$ are satisfied, and for almost all time steps $t \in I$ holds:*

$$\textit{Fluid momentum} \begin{cases} \hat{A}_1(\hat{U})(\hat{\psi}^v) = (\hat{J}\hat{\rho}_f \partial_t \hat{v}_f, \hat{\psi}^v)_{\widehat{\Omega}_f} + (\hat{\rho}_f \hat{J}(\widehat{F}^{-1}(\hat{v}_f - \hat{w}) \cdot \widehat{\nabla})\hat{v}_f), \hat{\psi}^v)_{\widehat{\Omega}_f} \\ +(\hat{J}\hat{\sigma}_f \widehat{F}^{-T}, \widehat{\nabla}\hat{\psi}^v)_{\widehat{\Omega}_f} - \langle \hat{g}_f, \hat{\psi}^v \rangle_{\widehat{\Gamma}_N} - (\hat{\rho}_f \hat{J} \hat{f}_f, \hat{\psi}^v)_{\widehat{\Omega}_f} \qquad = 0 \quad \forall \hat{\psi}^v \in \hat{V}_{f,\widehat{\Gamma}_i}^0, \end{cases}$$

$$\textit{Solid momentum, 1st eq.} \begin{cases} \hat{A}_2(\hat{U})(\hat{\psi}^v) = (\hat{\rho}_s \partial_t \hat{v}_s, \hat{\psi}^v)_{\widehat{\Omega}_s} + (\widehat{F}\widehat{\Sigma}, \widehat{\nabla}\hat{\psi}^v)_{\widehat{\Omega}_s} \\ +\gamma_w(\hat{v}_s, \hat{\psi}^v)_{\widehat{\Omega}_s} + \gamma_s(\hat{\epsilon}(\hat{v}_s), \widehat{\nabla}\hat{\psi}^v)_{\widehat{\Omega}_s} - (\hat{\rho}_s \hat{f}_s, \hat{\psi}^v)_{\widehat{\Omega}_s} \quad = 0 \quad \forall \hat{\psi}^v \in \hat{V}_s^0, \end{cases}$$

$$\textit{Fluid mesh motion} \left\{ \hat{A}_3(\hat{U})(\hat{\psi}^u) = (\hat{\sigma}_{mesh}, \widehat{\nabla}\hat{\psi}^u)_{\widehat{\Omega}_f} \quad = 0 \quad \forall \hat{\psi}^u \in \hat{V}_{f,\hat{u},\widehat{\Gamma}_i}^0, \right.$$

$$\textit{Solid momentum, 2nd eq.} \left\{ \hat{A}_4(\hat{U})(\hat{\psi}^u) = \hat{\rho}_s(\partial_t \hat{u}_s - \hat{v}_s, \hat{\psi}^u)_{\widehat{\Omega}_s} \quad = 0 \quad \forall \hat{\psi}^u \in \hat{L}_s, \right.$$

$$\textit{Fluid mass conservation} \left\{ \hat{A}_5(\hat{U})(\hat{\psi}^p) = (\widehat{div}(\hat{J}\widehat{F}^{-1}\hat{v}_f), \hat{\psi}^p)_{\widehat{\Omega}_f} \quad = 0 \quad \forall \hat{\psi}^p \in \hat{L}_f^0, \right.$$

$$\textit{Solid volume conservation} \left\{ \hat{A}_6(\hat{U})(\hat{\psi}^p) = (\hat{P}_s, \hat{\psi}^p)_{\widehat{\Omega}_s} \quad = 0 \quad \forall \hat{\psi}^p \in \hat{L}_s^0. \right.$$

The Neumann coupling conditions on $\widehat{\Gamma}_i$ are fulfilled in a variational way and cancel in monolithic modeling. Thus, the condition

$$\langle \hat{J}\hat{\sigma}_f \widehat{F}^{-T} \hat{n}_f, \hat{\psi}_f^v \rangle_{\widehat{\Gamma}_i} + \langle [\widehat{F}\widehat{\Sigma}]\hat{n}_s, \hat{\psi}_s^v \rangle_{\widehat{\Gamma}_i} = 0 \quad \forall \hat{\psi}_f^v \in H^1(\widehat{\Omega}_f),\ \forall \hat{\psi}_s^v \in H^1(\widehat{\Omega}_s), \quad \hat{\psi}_f^v = \hat{\psi}_s^v \quad \text{on } \widehat{\Gamma}_i, \tag{337}$$

is implicitly contained in the above system.

14.9.1 A compact semi-linear form

Consequently, we can write in short form:

Problem 14.13. *Find $\hat{U} \in \widehat{X}_0^D$ such that $\hat{v}_f(0) = \hat{v}_f^0$, $\hat{v}_s(0) = \hat{v}_s^0$, $\hat{u}_f(0) = \hat{u}_f^0$, and $\hat{u}_s(0) = \hat{u}_s^0$ are satisfied, and for almost all time steps $t \in I$ holds:*

$$\hat{A}(\hat{U})(\hat{\Psi}) = 0 \quad \textit{for all } \hat{\Psi} \in \widehat{X} \tag{338}$$

with $\hat{\Psi} = \{\hat{\psi}_f^v, \hat{\psi}_s^v, \hat{\psi}_f^u, \hat{\psi}_s^u, \hat{\psi}_f^p, \hat{\psi}_s^p\}$ and $\widehat{X} = \hat{V}_{f,\hat{v}}^0 \times \hat{L}_f \times \hat{V}_{f,\hat{u},\widehat{\Gamma}_i}^0 \times \hat{V}_s^0 \times \hat{L}_f^0 \times \hat{L}_s^0$ and

$$\hat{A}(\hat{U})(\hat{\Psi}) := \hat{A}_1(\hat{U})(\hat{\psi}^v) + \hat{A}_2(\hat{U})(\hat{\psi}^v) + \hat{A}_3(\hat{U})(\hat{\psi}^u) + \hat{A}_4(\hat{U})(\hat{\psi}^u) + \hat{A}_5(\hat{U})(\hat{\psi}^p) + \hat{A}_6(\hat{U})(\hat{\psi}^p).$$

We apply the previous ideas to fluid-structure interaction. Here the abstract problem is formally given by:

Definition 14.14. *Find* $\hat{U} = \{\hat{v}_f, \hat{v}_s, \hat{u}_f, \hat{u}_s, \hat{p}_f, \hat{p}_s\} \in \widehat{X}_D^0$, *where* $\widehat{X}_D^0 := \{\hat{v}_f^D + \hat{V}_{f,\hat{v}}^0\} \times \hat{L}_f \times \{\hat{u}_f^D + \hat{V}_{f,\hat{u}}^0\} \times \{\hat{u}_s^D + \hat{V}_s^0\} \times \hat{L}_f^0 \times \hat{L}_s^0$, *such that*

$$\overline{\hat{A}}(\hat{U})(\hat{\Psi}) = \int_0^T \hat{A}(\hat{U})(\hat{\Psi})\, dt = 0 \quad \forall \hat{\Psi} \in \widehat{X}, \tag{339}$$

where $\hat{\Psi} = \{\hat{\psi}_f^v, \hat{\psi}_s^v, \hat{\psi}_f^u, \hat{\psi}_s^u, \hat{\psi}_f^p, \hat{\psi}_s^p\}$ *and* $\widehat{X} = \hat{V}_{f,\hat{v}}^0 \times \hat{L}_f \times \hat{V}_{f,\hat{u},\widehat{\Gamma}_i}^0 \times \hat{V}_s^0 \times \hat{L}_f^0 \times \hat{L}_s^0$ *and*

$$\hat{A}(\hat{U})(\hat{\Psi}) := \hat{A}_1(\hat{U})(\hat{\psi}^v) + \hat{A}_2(\hat{U})(\hat{\psi}^v) + \hat{A}_3(\hat{U})(\hat{\psi}^u) + \hat{A}_4(\hat{U})(\hat{\psi}^u) + \hat{A}_5(\hat{U})(\hat{\psi}^p) + \hat{A}_6(\hat{U})(\hat{\psi}^p).$$

as defined in Problem 14.12.

14.9.2 Preparing temporal discretization: grouping the FSI problem

Definition 14.15 (Arranging of the FSI semi-linear form into groups)**.** *We formally define the following semi-linear forms and group them into four categories:*

- *time equation terms (including the time derivatives);*
- *implicit terms (e.g., the incompressibility of the fluid);*
- *pressure terms;*
- *all remaining terms (stress terms, convection, damping, etc.);*

such that

$$\hat{A}_T(\hat{U})(\hat{\Psi}) = (\hat{J}\hat{\rho}_f \partial_t \hat{v}_f, \hat{\psi}_f^v)_{\widehat{\Omega}_f} - (\hat{\rho}_f \hat{J}(\widehat{F}^{-1}\hat{w}\cdot\widehat{\nabla})\hat{v}_f), \hat{\psi}_f^v)_{\widehat{\Omega}_f} + (\hat{\rho}_s \partial_t \hat{v}_s, \hat{\psi}_s^v)_{\widehat{\Omega}_s} + (\hat{\rho}_s \partial_t \hat{u}_s, \hat{\psi}_s^u)_{\widehat{\Omega}_s}, \tag{340}$$

$$\hat{A}_I(\hat{U})(\hat{\Psi}) = (\alpha_u \widehat{\nabla}\hat{u}_f, \widehat{\nabla}\hat{\psi}_f^u)_{\widehat{\Omega}_f} + (\widehat{div}(\hat{J}\widehat{F}^{-1}\hat{v}_f), \hat{\psi}_f^p)_{\widehat{\Omega}_f} + (\hat{P}_s, \hat{\psi}_s^p)_{\widehat{\Omega}_s}, \tag{341}$$

$$\hat{A}_E(\hat{U})(\hat{\Psi}) = (\hat{\rho}_f \hat{J}(\widehat{F}^{-1}\hat{v}_f \cdot \widehat{\nabla})\hat{v}_f), \hat{\psi}_f^v)_{\widehat{\Omega}_f} + (\hat{J}\widehat{\sigma}_{f,vu}\widehat{F}^{-T}, \widehat{\nabla}\hat{\psi}_f^v)_{\widehat{\Omega}_f} - \langle \rho_f \nu \widehat{F}^{-T}\widehat{\nabla} v^T \hat{n}, \hat{\psi}_f^v\rangle_{\widehat{\Gamma}_{out}} \tag{342}$$

$$+ (\widehat{F}\widehat{\Sigma}, \widehat{\nabla}\hat{\psi}_s^v)_{\widehat{\Omega}_s} - (\hat{\rho}_s \hat{v}_s, \hat{\psi}_s^u)_{\widehat{\Omega}_s}, \tag{343}$$

$$\hat{A}_P(\hat{U})(\hat{\Psi}) = (\hat{J}\widehat{\sigma}_{f,p}\widehat{F}^{-T}, \widehat{\nabla}\hat{\psi}_f^v)_{\widehat{\Omega}_f} + (\hat{J}\widehat{\sigma}_{s,p}\widehat{F}^{-T}, \widehat{\nabla}\hat{\psi}_s^v)_{\widehat{\Omega}_s}, \tag{344}$$

where the reduced stress tensors $\widehat{\sigma}_{f,vu}$, $\widehat{\sigma}_{f,p}$, *and* $\widehat{\sigma}_{s,p}$ *are defined as:*

$$\widehat{\sigma}_{f,p} = -\hat{p}_f \hat{I}, \quad \widehat{\sigma}_{f,vu} = \rho_f \nu_f (\widehat{\nabla}\hat{v}_f \widehat{F}^{-1} + \widehat{F}^{-T}\widehat{\nabla}\hat{v}_f^T), \tag{345}$$

$$\widehat{\sigma}_{s,p} = -\hat{p}_s \hat{I}, \quad \textit{(if we deal with the INH or IMR material)}, \tag{346}$$

and $\widehat{\Sigma}$ *denotes as usual the structure tensor of the INH, IMR, or STVK material. The time derivative in* $\hat{A}_T(\hat{U})(\hat{\Psi})$ *is approximated by a backward difference quotient. For the time step* $t_n \in I$ *for* $n = 1, 2, \ldots, N$ *(*$N \in \mathbb{R}$*), we compute* $\hat{v}_i := \hat{v}_i^n, \hat{u}_i := \hat{u}_i^n$ $(i = f, s)$ *via*

$$\hat{A}_T(\hat{U}^{n,k})(\hat{\Psi}) \approx \frac{1}{k}\big(\hat{\rho}_f \hat{J}^{n,\theta}(\hat{v}_f - \hat{v}_f^{n-1}), \hat{\psi}^v\big)_{\widehat{\Omega}_f} - \frac{1}{k}\big(\hat{\rho}_f(\hat{J}\widehat{F}^{-1}(\hat{u}_f - \hat{u}_f^{n-1})\cdot\widehat{\nabla})\hat{v}_f, \hat{\psi}^v\big)_{\widehat{\Omega}_f} \tag{347}$$

$$+ \frac{1}{k}\big(\hat{\rho}_s(\hat{v}_s - \hat{v}_s^{n-1}), \hat{\psi}^v\big)_{\widehat{\Omega}_s} + \big(\hat{u}_s - \hat{u}_s^{n-1}, \hat{\psi}^u\big)_{\widehat{\Omega}_s}, \tag{348}$$

where we introduce a parameter θ, *which is clarified below. Furthermore, we use*

$$\hat{J}^{n,\theta} = \theta \hat{J}^n + (1-\theta)\hat{J}^{n-1},$$

and $\hat{u}_i^n := \hat{u}_i(t_n)$, $\hat{v}_i^n := \hat{v}_i(t_n)$, *and* $\hat{J} := \hat{J}^n := \hat{J}(t_n)$. *The former time step is given by* $\hat{v}_i^{n-1}$, *etc. for* $i = f, s$.

Remark 14.16. *Working with damped solid equations, one needs to add two terms such that:*

$$\hat{A}_E(\hat{U})(\hat{\Psi}) = \ldots + \gamma_w(\hat{v}_s, \hat{\psi}_s^v)_{\widehat{\Omega}_s} + \gamma_s(\hat{\epsilon}(\hat{v}_s), \widehat{\nabla}\hat{\psi}_s^v)_{\widehat{\Omega}_s}.$$

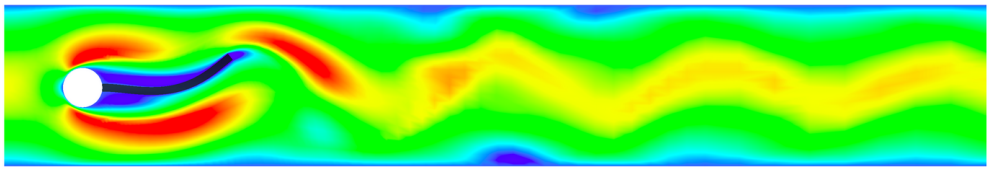
14.9.3 Implementation in ANS/deal.II and DOpElib; both based on C++

To supplement our algorithmic discussion with some practical aspects, the reader is invited to test him/her-self some concepts presented so far. Implementations are present in two software packages deal.II [10] (source code [137] in ANS[14]) and DOpElib [41, 62]:

- ALE-FSI in deal.II (ANS) with biharmonic mesh motion
  https://journals.ub.uni-heidelberg.de/index.php/ans/issue/view/1244 [137] solving FSI benchmark problems [77]
- Nonstationary Navier-Stokes benchmark problems [118] at
  http://www.dopelib.uni-hamburg.de/ `Examples/PDE/InstatPDE/Example1`
- ALE-FSI benchmark problems [77] with biharmonic mesh motion at
  http://www.dopelib.uni-hamburg.de/ `Examples/PDE/InstatPDE/Example2`
- Biot-Lame-Navier system: augmented Mandel's benchmark at
  http://www.dopelib.uni-hamburg.de/ `Examples/PDE/InstatPDE/Example6`
- Stationary FSI optimization at
  http://www.dopelib.uni-hamburg.de/ `Examples/OPT/StatPDE/Example9`

We notice that the implementation is performed in a *practical* monolithic way as described in Chapter 14 that has two assumptions; namely, displacements and velocity are taken from globally-defined Sobolev spaces rather than restricting them to the sub-domains and one (major) limitation (the development of an iterative linear solver and preconditioners might be challanging since we do not distuingish the sub-problems in the system matrix). Putting these two limitations aside, the idea results in a fascinating easy way to implement multiphysics problems in non-overlapping domains.

Remark 14.17. *It has been appeared that the basic program structure could be easily generalized and extended to other complex multiphysics problems such as moving boundary problems with chemical reactions [88] as well as phase-field fracture propagation [94, 139]* ◇



Archive of Numerical Software 1(1), 1-19, 2013

Solving Monolithic Fluid-Structure Interaction Problems in Arbitrary Lagrangian Eulerian Coordinates with the deal.II Library

Thomas Wick

We describe a setting of a nonlinear fluid-structure interaction problem and the corresponding solution process in the finite element software package deal.II. The fluid equations are transformed via the ALE mapping (Arbitrary Lagrangian Eulerian framework) to a reference configuration and these are coupled with the structure equations by a monolithic solution algorithm. To construct the ALE mapping, we use a biharmonic equation. Finite differences are used for temporal discretization. The derivation is realized in a general manner that serves for different time stepping schemes. Spatial discretization is based on a Galerkin finite element scheme. The nonlinear system is solved by a Newton method. Using this approach, the Jacobian matrix is constructed by exact computation of the directional derivatives. The implementation using the software library package deal.II serves for the computation of different fluid-structure configurations. Specifically, our geometry data are taken from the fluid-structure benchmark configuration that was proposed in 2006 in the DFG project Fluid-Structure Interaction I: Modelling, Simulation, Optimisation. Our results show that this implementation using deal.II is able to produce comparable findings.

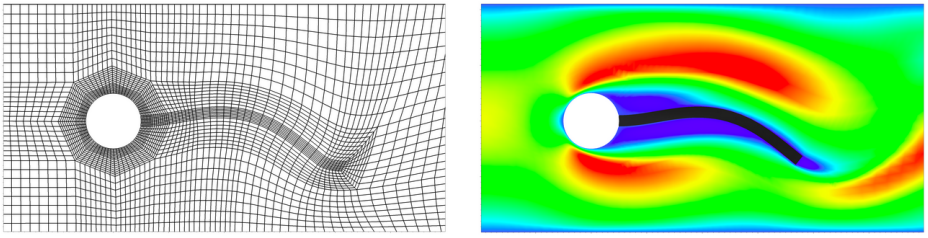


Figure 68: Reference [137] at https://media.archnumsoft.org/10305/

[14] ANS = Archive of Numerical Software, http://www.archnumsoft.org/

The structure of such a program for time-dependent nonlinear problems is as follows:

- Input of all data: geometry, material parameters, right hand side values
- Sorting and associating degrees of freedom
- Assembling the Jacobian
- Assembling the right hand side residual
- Newton's method
- Solution of linear equations
- Postprocessing (output, a posteriori error estimation, mesh refinement)
- Main routine including time step loop

14.10 Diffusive interface approximations

Level set method: (Osher S., Sethian J.A., Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations (Journal of Computational Physics, 79(1), page 12-49, 1988).

Phase-field methods: Cahn-Hilliard, potential, ...

14.11 Sharp interface approximations in cut-elements with a locally modified FEM

Originally proposed in [53, 54] with an open-source code implementation in deal.II on https://doi.org/10.5281/zenodo.1457758.

In this method, the interface can cut the elements, but is not reconstructed in a diffusive manner, but in an exact fashion. The key trick is not to introduce new degrees of freedom on the cuts, but to design a patch and to move the existing degrees of freedom to the cuts. With this, the inner structure of the system matrix and vectors remain the same and do not change at each time step in a time-dependent computation. Thus, the setup of numerical solvers is greatly facilitated.

14.12 On the nonlinear and linear numerical solutions

Coupled problems can be solved with methods discussed in Chapter 13. While fixed-point or Newton methods can be relatively easily applied, the crucial point are the linear equation systems inside Newton's method. For multiple PDEs with many degrees of freedom, direct solvers such as UMFPACK or MUMPS are too costly and memory-demanding (see lectures on Numerik 1). Consequently, we need iterative solvers, which require (nearly) always a preconditioner. To construct such preconditioners is problem-specific and must be re-done for each coupled PDE system. Therefore, this is a lot of work and very time-consuming. However, for an efficient numerical solution, there are only a few alternatives. Black-box preconditioners may work in certain cases. Another possibility is to decouple the PDEs and to iterate (staggered/partitioned approach) and to use well-known solvers for the subproblems.

Either way, in coupled problems, often material and model parametes differ in several order of magnitude. Therefore, nonlinear and linear solvers may suffer from ill-conditioning and it must be proofed (numerical analysis!) that they are robust under parameter variations. This equally belongs to the development of good solvers.

14.12.1 Example

Our example of an iterative solver, namely GMRES with Schur complements and multigrid preconditioning was developed and presented in

- Monolithic solver with emphasis on Schur complements [81]
- Parallel monolithic solver [80]

In the latter paper, an extensive literature overview for solvers for fluid-structure interaction as of the year 2018 is provided.

14.13 Chapter summary and outlook

We introduced concepts to couple two (or more) PDEs. Many parts of this section are also current research topics. Many open problems can be found in numerical analysis (convergence order), well-posedness, robust and efficient numerical methods, efficient solvers, and so forth.

15 Public climate school (November 2019/2021): PDEs in atmospheric physics

The underlying main literature is the textbook [49]. The introduction to weather models in [121][Chapter 2] is highly recommended, too. In meteorology, mainly six PDEs are coupled, which can be simplified by certain assumptions. However, for the latter we refer to the corresponding literature [49, 121] and references cited therein. We mainly follow now [49]. In the following, we first formulate the PDEs, then classify them, and focus lastly on the Navier-Stokes equations.
Disclaimer: The following models are taken in such a way that they still make sense from a physics point of view and serve for our numerical explanations (the main goal in this class!). For their 'correct' behavior in practice and further modifications and/or simplifications, we do not claim full knowledge and refer to the previously given references.

15.1 Notation

Let $\Omega \subset \mathbb{R}^d, d = 3$ the spatial domain and $I := (0, T]$ the time interval.

15.2 Five governing equations in meteorology

To model meteorological events and air flow, basically five variables are important:

- air pressure $p : \Omega \times I \to \mathbb{R}$;
- air density $\rho : \Omega \times I \to \mathbb{R}$;
- Humidity of the air $w : \Omega \times I \to \mathbb{R}$;
- air temperature $T : \Omega \times I \to \mathbb{R}$;
- wind velocities $v : \Omega \times I \to \mathbb{R}^d$.

For these variables, we need five governing equations. These are described in the following.

Formulation 15.1 (Ideal gas equation for p)**.** *Let $R > 0$ be the gas constant (see e.g., [49][Chapter 2]), $\rho : \Omega \times I \to \mathbb{R}$ the air density, $T : \Omega \times I \to \mathbb{R}$ the temperature. Find $p : \Omega \times I \to \mathbb{R}$ such that*

$$\begin{aligned} p &= \rho R T \quad \text{in } \Omega \times I, \\ \text{plus bc} &\quad \text{on } \partial\Omega \times I. \end{aligned}$$

Formulation 15.2 (Continuity equation / mass conservation for ρ)**.** *Let $v : \Omega \times I \to \mathbb{R}^d$ be the air velocity. Find $\rho : \Omega \times I \to \mathbb{R}$ such that*

$$\begin{aligned} \partial_t \rho + \nabla \cdot (\rho v) &= 0 \quad \text{in } \Omega \times I, \\ \text{plus bc} &\quad \text{on } \partial\Omega \times I, \\ \rho(0) &= \rho_0 \quad \text{in } \Omega \times \{t = 0\}. \end{aligned}$$

Remark 15.3. *Compare this to the law in the incompressible NSE from the chapters before. Specifically, when $\rho = const$, then we obtain $\nabla \cdot v = 0$.*

The next law is also a continuity equation, but now for the air water steem; see [49][Chapter 4] or [121][Eq. (2.5)] to describe a single phase concentration (e.g., gaseous water phase):

Formulation 15.4 (Humidity for w; concentration of gaseous water phase)**.** *Let $W := W(T, w, \rho) \in \mathbb{R}$ quantify the processes that govern phase transitions. Let $v : \Omega \to \mathbb{R}$ be the air velocity. Find $w : \Omega \times I \to \mathbb{R}$ such that*

$$\begin{aligned} \partial_t w + \nabla \cdot (wv) &= W \quad \text{in } \Omega \times I, \\ \text{plus bc} &\quad \text{on } \partial\Omega \times I, \\ w(0) &= w_0 \quad \text{in } \Omega \times \{t = 0\}. \end{aligned}$$

For the temperature, we adopt the first law of thermodynamics (see [49][Chapter 3]):

$$dE = dQ - dA,$$

which is the conservation of energy. Here, dE is the inner energy, dQ the heat applied to the system, dA the work performed by the system. One possible realization in terms of continuum mechanics is then [49][page 52, (5.3)]: Let m be the mass and specific heat capacity $c_p > 0$ and δQ the net heat flow (energy flow). Find $T : \Omega \times I \to \mathbb{R}$ such that

$$mc_p \partial_t T = mv\partial_t p + \frac{d}{dt}\delta Q \quad \text{in } \Omega \times I.$$

After some further manipulations, one arrives at [121]

Formulation 15.5 (First law of thermodynamics for T)**.** *Given the specific heat capacity $c_p > 0$, and thermal conductivity $k > 0$. Furthermore, let the density ρ, the pressure p and the velocity v be determined by the continuity equation, ideal gas law and the Navier-Stokes equations, respectively. Let F be the net radiative flux (amount of power radiated through a given area) and $\frac{d}{dt}\delta Q := \frac{d}{dt}\delta Q(T, p, \rho)$ as before the rate of internal heating/cooling, for instance due to phase changes in atmosspheric moisture. Find $T : \Omega \times I \to \mathbb{R}$ such that*

$$\begin{aligned} \rho c_p \partial_t T + p\nabla \cdot v - \nabla \cdot (k\nabla T) &= -\nabla \cdot F + \rho \frac{d}{dt}\delta Q \quad \text{in } \Omega \times I, \\ T &= T_D \quad \text{on } \partial\Omega \times I, \\ T(0) &= T_0 \quad \text{in } \Omega \times \{t = 0\}, \\ p(0) &= p_0 \quad \text{in } \Omega \times \{t = 0\}, \end{aligned}$$

where the Dirichlet conditions $T = T_D$ are only an example. Neumann conditions are equally important.

Finally, we describe the air flow. Since the earth turns, we have to include Coriolis forces (which is a virtual force) and due to the earth rotation. Consequently, the equations look a bit different to our previous chapters. We now simply take [49][Chapter 18, page 259, (18.10)]

Formulation 15.6 (Navier-Stokes for v)**.** *Given the angle velocity ω, the Coriolis force $2\omega \times v$, the gravitational potential Φ, the density ρ and the pressure p, the dynamic viscosity μ. Find $v : \Omega \times I \to \mathbb{R}^d$ such that*

$$\begin{aligned} \partial_t v + (v \cdot \nabla)v - \frac{\mu}{\rho}[\Delta v + \frac{1}{3}\nabla(\nabla \cdot v)] + 2\omega \times v + \frac{1}{\rho}\nabla p &= -\nabla\Phi \quad \text{in } \Omega \times I, \\ v &= v_D \quad \text{on } \partial\Omega \times I, \\ v(0) &= v_0 \quad \text{in } \Omega \times \{t = 0\}, \end{aligned}$$

where v_D (Dirichlet conditions) are just an example. Neumann conditions are equally important.

Remark 15.7. *We briefly state as comparison from Section 13.1.4 a classical flow model using the isothermal incompressible Navier-Stokes formulation: Find $v : \Omega \times I \to \mathbb{R}^n$ and $p : \Omega \times I \to \mathbb{R}$*

$$\begin{aligned} \rho\partial_t v + \rho(v \cdot \nabla)v - \mu\Delta v + \nabla p &= \rho f \qquad \textit{(related to Formulation (15.6))} \\ \nabla \cdot v &= 0 \qquad \textit{(related to Formulation (15.2))} \end{aligned}$$

with $\mu = \rho\nu$, where ν is the kinematic viscosity.

15.3 Some values and units

We illustrate the previous variables with some specific values. For the unit of the pressure, we have

$$[p] = Pa = \frac{N}{m^2} = \frac{kg}{ms^2} = \frac{J}{m^3}$$

The atmospheric air pressure close to the earth surface is

$$p_0 = 101325Pa = 1.01325bar.$$

Example 15.8. *In the daily weather forecast we see high pressure regions (Hochdruckgebiet) and low pressure regions (Tiefdruckgebiet). They range between* 1030 hPa *for high pressure and* 970 hPa *for low pressure.*

Example 15.9. *On the moon the atmospheric air pressure close to the surface is* $3 \cdot 10^{-10} hPa$ *and the pressure in a bike tire is between* $4 - 8$ $bar = 4000 - 8000$ hPa.

The specific gas constant is

$$[R] = 287.058 \frac{J}{kg \cdot K}$$

where K stands for Kelvin

$$K = Celsius + 273.15$$

The unit for the air density is

$$[\rho] = \frac{kg}{m^3}$$

The temperature is measured in

$$[T] = K$$

The velocity is measured in

$$[v] = \frac{m}{s}.$$

Usually the humidity is measured as

$$[w] = \frac{g}{m^3},$$

which however is not an SI unit. Consider $1g = 10^{-3} kg$.
Energy:

$$[dE] = J = N \cdot m.$$

Energy flow or power (mechanical work of $1J$ per second):

$$[d/dt \delta Q] = 1Js^{-1} = 1W = \frac{Nm}{s}$$

Net radiative flux:

$$[F] = W/m^2$$

Example 15.10. *In the year* 2021 *(first half year), the total power of wind energy in Germany was* 55 772 MW, *approx.* 56 GW, *where* $W = Watt$.

15.4 Classifications

For the terminology, we refer to Chapter 5.

15.4.1 Conservation laws

We derived the previous equations based on physical conservation laws of mass, momentum and energy. Furthermore conservation of water phases of water in the solid, liquid, and gaseous phases may enter [121][Section 2.1.1]. Constituent aerosal concentrations can also be quantified by conservations relations [121][Section 2.1.1]. Therefore (we repeat ourselves), physics-based discretization schemes in time and space are important.

15.4.2 Order, steady-state vs. nonstationary, single vs. vector-valued

- Pressure: zero order (algebraic equation), single equation
- Density: first order in space and time, single equation
- Water stem: first order in space and time, single equation
- Temperature: first order in time, single equation
- Velocities: second order in space, first order in time, vector-valued equation

15.4.3 Coupling and nonlinearities

The previous governing equations are clearly coupled: the variables from the single equations enter into the other ones. It is obvious that

- The Navier-Stokes equations are nonlinear (semi-linear) because of the convection term $(v \cdot \nabla)v$;
- All other equations are linear, when the solution variables from the other equations are assumed to be given.
- If two equations are solved simultaneously (monolithically), then they may become nonlinear because of their coupling. For instance, if equation No. 2 and 5 are solved simultaneously than ρ and v are solution variables. Consequently equation No. 2 is nonlinear due to $\rho\nabla \cdot v$.
- Not all solution variables enter into all other equations

15.4.4 Identification of similar structures

Overall, the previous classifications allow to identify similar structures. For instance the first law of thermodynamics and the Navier-Stokes equations share similar structures that allow for similar numerical discretizations:

$$\begin{aligned} \rho c_p \partial_t T \quad &\text{and} \quad \rho \partial_t v \quad \text{(temporal derivatives)} \\ -\nabla \cdot (k \nabla T) \quad &\text{and} \quad -\nabla \cdot (\nu \nabla v) \quad \text{(diffusive terms)} \\ \nabla \cdot (\rho v) \quad &\text{and} \quad \nabla \cdot (w v) \quad \text{(divergence terms)} \end{aligned}$$

As well, the continuity equation (mass conservation) has the same time derivative and also a convective behavior:

$$\partial_t \rho, \quad \nabla \cdot (\rho v),$$

but no diffusion (which is an important difference resulting in a different order of the PDE). Finally, the Navier-Stokes equations and the temperature equation are basically of parabolic type and can be related to the prototype heat equation $\partial_t u - \Delta u = f$.

15.5 Weak formulations and solution algorithms

15.5.1 Examples of weak formulations

Construct first the function spaces. For instance V^ρ. Then:

$$\begin{aligned} &\partial_t \rho + \rho \nabla \cdot v = 0 \\ &\quad \Rightarrow (\partial_t \rho, \psi^\rho) + (\rho \nabla \cdot v, \psi^\rho) =: A(\rho)(\psi^\rho). \end{aligned}$$

For the Navier-Stokes equations, we obtain:

$$\begin{aligned} &\partial_t v + (v \cdot \nabla)v - \frac{\mu}{\rho}[\Delta v + \frac{1}{3}\nabla(\nabla \cdot v)] + 2\omega \times v = -\frac{1}{\rho}\nabla p - \nabla \Phi, \\ &\Rightarrow \underbrace{(\partial_t v, \psi^v) + ((v \cdot \nabla)v, \psi^v) + (\frac{\mu}{\rho}\nabla v, \nabla \psi^v) + (\frac{\mu}{\rho}\frac{1}{3}(\nabla \cdot v)I, \nabla \psi^v) + (2\omega \times v, \psi^v)}_{=A(v)(\psi^v)} = \underbrace{-(\frac{1}{\rho}\nabla p, \psi^v) - (\nabla \Phi, \psi^v)}_{=F(\psi^v)} \end{aligned}$$

Here, in the second order spatial terms, we applied integration by parts. Specifically, we want to mention:

$$\begin{aligned} -\nabla \cdot (\nabla v) \quad &\rightarrow \quad (-\nabla \cdot (\nabla v), \psi) = (\nabla v, \nabla \psi) \quad \text{(if } v = 0 \text{ on } \partial\Omega\text{; Green's formula; well-known)} \\ -\nabla(\nabla \cdot v) \quad &\rightarrow \quad (-\nabla(\nabla \cdot v), \psi) = ((\nabla \cdot v)I, \nabla \psi) \quad \text{(if } v = 0 \text{ on } \partial\Omega\text{; a bit less well-known)} \end{aligned}$$

Proposition: It holds for $u \in H^1(\Omega)$

$$\underbrace{(\nabla(\nabla\cdot u)}_{(*)}, \varphi) = (\nabla\cdot u\, \mathbb{J}, \nabla\varphi) \quad \text{for } \varphi \in H^1_0(\Omega)$$

Proof. (2D)

$\Rightarrow$ component-wise:

$$(*) = \left(\nabla\left(\partial_x u_x + \partial_y u_y\right), \begin{pmatrix}\varphi_x \\ \varphi_y\end{pmatrix}\right)$$

$$= \left(\begin{pmatrix}\partial_x(\partial_x u_x + \partial_y u_y) \\ \partial_y(\partial_x u_x + \partial_y u_y)\end{pmatrix}, \begin{pmatrix}\varphi_x \\ \varphi_y\end{pmatrix}\right)$$

$$= \int \partial_{xx} u_x + \partial_x\partial_y u_y\, \varphi_x + \partial_y\partial_x u_x + \partial_{yy} u_y\, \varphi_y \, d\vec{x}$$

$$= -\int_{\partial\Omega} \Big(\underbrace{[\partial_x u_x + \partial_y u_y]}_{=\nabla\cdot u} \varphi_x n_x + \underbrace{[\partial_x u_x + \partial_y u_y]}_{=\nabla\cdot u} \varphi_y n_y \Big) d\vec{s}$$

$$- \int_\Omega \Big(\underbrace{[\partial_x u_x + \partial_y u_y]}_{=\nabla\cdot u} \partial_x \varphi_x + \underbrace{[\partial_x u_x + \partial_y u_y]}_{=\nabla\cdot u} \partial_y \varphi_y \Big) d\vec{x}$$

$$= -\int_{\partial\Omega} \nabla\cdot u \underbrace{\begin{pmatrix}1 & 0 \\ 0 & 1\end{pmatrix}}_{=\mathbb{J}} \begin{pmatrix}\varphi_x \\ \varphi_y\end{pmatrix} \begin{pmatrix}n_x \\ n_y\end{pmatrix} d\vec{s}$$

$$- \int_\Omega \nabla\cdot u\, \mathbb{J} : \begin{pmatrix}\partial_x\varphi_x & 0 \\ 0 & \partial_y\varphi_y\end{pmatrix} d\vec{x}$$

(Frobenius; $= \nabla\varphi$)

$$= -\int_{\partial\Omega} \cdots - \int_\Omega \nabla\cdot u\, \mathbb{J} : \begin{pmatrix}\partial_x\varphi_x & \partial_y\varphi_x \\ \partial_x\varphi_y & \partial_y\varphi_y\end{pmatrix} d\vec{x}$$

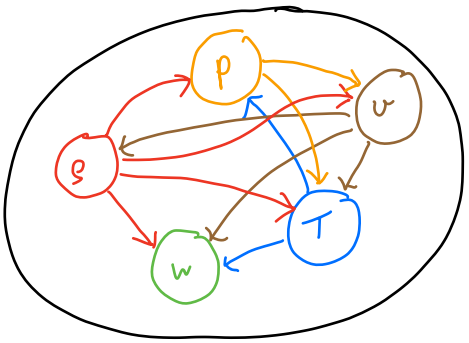
$$= -\int_{\partial\Omega} \nabla\cdot u\, \mathbb{J}\varphi n\, d\vec{s} - (\nabla\cdot u\, \mathbb{J}, \nabla\varphi)$$

(0 if $\varphi \in H^1_0(\Omega)$)

Recall:

$$(\nabla p, \varphi) = (-p\mathbb{J}, \nabla\varphi) - \int_{\partial\Omega} p\cdot n\varphi\, ds$$

15.5.2 Principle solution schemes: monolithic versus staggered



All-at-once solution
↳ monolithic scheme

BIG MATRIX

- costly to solve
- but no iteration error

$$\begin{pmatrix} * & * & \cdots \\ \vdots & & \end{pmatrix} \begin{pmatrix} p \\ \varrho \\ w \\ T \\ v \end{pmatrix} = \begin{pmatrix} f_p \\ 0 \\ f_w \\ f_T \\ f_v \end{pmatrix}$$

Alternative: staggered scheme; sequential order

- Smaller matrices (advantage)
- But iterations, which can be many and/or only conditionally stable
  ↓
  compare to explicit schemes in ODEs

15.5.3 A fully monolithic formulation

If all equations would be solved monolithically, all-at-once, (see Chapter 14), then the resulting system would be nonlinear and of the form:

Formulation 15.11. *Let X be the product function space, i.e., $X := V^p \times V^\rho \times V^w \times V^T \times V^v$ where V^p contain the possible Dirichlet boundary conditions. Find $U := (v, p, T, \rho, w) \in X$ such that $U(0) = U_0$ and for almost $t \in I$, it holds:*

$$A(U)(\Psi) = F(\Psi) \quad \forall \Psi \in X_0$$

with $\Psi := (\psi^v, \psi^p, \psi^T, \psi^\rho, \psi^w)$ and X_0 being the space in which all Dirichlet bc are set to zero. Furthermore, the compact semi-linear form is composed as

$$A(U)(\Psi) := A_p(p)(\psi^p) + A_\rho(\rho)(\psi^\rho) + A_w(w)(\psi^w) + A_T(T)(\psi^T) + A_v(v)(\psi^v)$$

where the single semi-linear forms represent the weak formulations of the single subproblems. Formally, a fixed-point or Newton-type solver can be applied from Chapter 13.

15.5.4 Staggered solution (time explicit coupling)

If the equations are solved sequentially or only partially fully-coupled, then iterative schemes must be constructed as explained in Section 14.4.

Algorithm 15.12. *Given the previous time step solution $U^{n-1} = (p^{n-1}, \rho^{n-1}, w^{n-1}, T^{n-1}, v^{n-1}) \in X$, for time step index $n = 1, \dots, N$ find $U^n = (p^n, \rho^n, w^n, T^n, v^n) \in X$ as follows:*

1. *Given $\rho^{n-1}, w^{n-1}, T^{n-1}, v^{n-1}$, find $p^n \in V^p : A_p(p)(\psi^p) = F_p(\psi^p)$ for all $\phi^p \in V^p$*
2. *Given $w^{n-1}, T^{n-1}, v^{n-1}, p^n$, find $\rho^n \in V^\rho : A_\rho(\rho)(\psi^\rho) = F_\rho(\psi^\rho)$ for all $\phi^\rho \in V^\rho$*
3. *Given $T^{n-1}, v^{n-1}, p^n, \rho^n$, find $w^n \in V^w : A_w(w)(\psi^w) = F_w(\psi^w)$ for all $\phi^w \in V^w$*
4. *Given $v^{n-1}, p^n, \rho^n, w^n$, find $T^n \in V^T : A_T(T)(\psi^T) = F_T(\psi^T)$ for all $\phi^T \in V^T$*
5. *Given p^n, ρ^n, w^n, T^n, find $v^n \in V^v : A_v(v)(\psi^v) = F_v(\psi^v)$ for all $\phi^v \in V^v$*

15.6 FEM discretization in 1D of ideal gas equation

We form the FEM discretization in 1D for Formulation 15.1. The strong form is given by

$$p = \rho R T \quad \text{in } \Omega,$$

where we notice that ρ and T are given, for instance as $\rho := \rho^{n-1}$ and $T := T^{n-1}$. We keep the boundary conditions open and simply choose $V^p := L^2(\Omega)$ as function space. Now we derive the weak form:

$$\begin{aligned} &\int_\Omega p\psi^p \, dx = \int_\Omega \rho R T \psi^p \, dx \\ &\Leftrightarrow (p, \psi^p) = (\rho R T, \psi^p) \end{aligned}$$

For the discretization, we choose the finite dimensional space $V_h^p = \{\psi_1^p, \dots, \psi_S^p$ such that $\dim(V_h^p) = S$. Then, we have: Find $p_h \in V_h^p$ such that

$$(p_h, \psi_h^p) = (\rho_h R T_h, \psi_h^p) \quad \forall \psi_h^p \in V_h^p.$$

Or using the abstract notation: Find $p_h \in V_h^p$ such that

$$a(p_h, \psi_h^p) = l(\psi_h^p) \quad \forall \psi_h^p \in V_h^p.$$

We now choose the linear combination

$$p_h = \sum_{i=1}^{S} \chi_i^p \psi_i^p$$

and insert

$$\sum_{i=1}^{S} \chi_i^p(\psi_i^p, \psi_h^p) = l(\psi_h^p) \quad \forall \psi_h^p \in V_h^p.$$

Since this holds for all test functions, we have

$$\sum_{i=1}^{S} \chi_i^p(\psi_i^p, \psi_j^p) = l(\psi_j^p) \quad \forall \psi_j^p \in V_j^p.$$

Finally, use employ P_1 hat functions (see Section 8.4) and obtain:

$$\begin{aligned}
j = i : (\psi_i^p, \psi_i^p) &= \int_{x_{i-1}}^{x_{i+1}} \psi_i^p \psi_i^p \, dx = \frac{2}{3} h \\
j = i-1 : (\psi_i^p, \psi_{i-1}^p) &= \int_{x_{i-1}}^{x_i} \psi_i^p \psi_{i-1}^p \, dx = \frac{1}{6} h \\
j = i+1 : (\psi_i^p, \psi_{i+1}^p) &= \int_{x_i}^{x_{i+1}} \psi_i^p \psi_{i+1}^p \, dx = \frac{1}{6} h \\
j = i-2, \ldots, (\psi_i^p, \psi_{i+1}^p) &= 0 \\
j = i+2, \ldots, (\psi_i^p, \psi_{i+1}^p) &= 0.
\end{aligned}$$

Consequently, the system matrix reads

$$M = \frac{h}{6} \begin{pmatrix} 4 & 1 & & & 0 \\ 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 4 & 1 \\ 0 & & & 1 & 4 \end{pmatrix}.$$

15.7 Navier-Stokes discretization

For details on stationary Navier-Stokes, we refer the reader to Section 13.19. In the nonstationary case, we first need to discretize time and then obtain a sequence of quasi-stationary problems; see Section 12.4. Moreover, we refer to the textbook [132].

15.8 Boussinesq approximation

This is an extension to the pure incompressible Navier-Stokes equations in meteorology. We refer the reader to [49][Chapter 12.7]. In the presense of turbulence phenomena, we again refer to [49] or as well to [127].

16 Computational convergence analysis

We provide some tools to perform a computational convergence analysis. In these notes we faced two situations of 'convergence':

- **Discretization error:** Convergence of the discrete solution u_h towards the (unknown) exact solution u;
- **Iteration error:** Convergence of an iterative scheme to approximate the discrete solution u_h through a sequence of approximate solutions $u_h^{(k)}, k = 1, 2, \ldots$.

In the following we further illustrate the terminologies 'first order convergence', 'convergene of order two', 'quadratic convergence', 'linear convergence', etc.

16.1 Discretization error

Before we go into detail, we discuss the relationship between the degrees of freedom (DoFs) N and the mesh size parameter h. In most cases the discretization error is measured in terms of h and all a priori and a posteriori error estimates are stated in a form

$$\|u - u_h\| = O(h^\alpha), \quad \alpha > 0.$$

In some situations it is however better to create convergence plots in terms of DoFs vs. the error. One example is when adaptive schemes are employed with different h. Then it would be not clear to which h the convergence plot should be drawn. But simply counting the total numbers of DoFs is not a problem though.

16.1.1 Relationship between h and N (DoFs)

The relationship of h and N depends on the basis functions (linear, quadratic), whether a Lagrange method (only nodal points) or Hermite-type method (with derivative information) is employed. Moreover, the dimension of the problem plays a role.

We illustrate the relationship for a Lagrange method with linear basis functions in 1D,2D,3D:

Proposition 16.1. *Let d be the dimension of the problem: $d = 1, 2, 3$. It holds*

$$N = \left(\frac{1}{h} + 1\right)^d$$

where h is the mesh size parameter (lengh of an element or diameter in higher dimensions for instance), and N the number of DoFs.

Proof. Sketch. No strict mathematical proof. We initialize as follows:

- 1D: 2 values per line;
- 2D: 4 values per quadrilaterals;
- 3D: 8 values per hexahedra.

Of course, for triangles or primsms, we have different values in 2D and 3D. We work on the unit cell with $h = 1$. All other h can be realized by just normalizing h. By simple counting the nodal values, we have in 1D

```
h    N
=======
1    2
1/2  3
1/4  5
1/8  9
1/16 17
1/32 33
...
=======
```

We have in 2D

```
h    N
=======
1    4
1/2  9
1/4  25
1/8  36
1/16 49
1/32 64
...
=======
```

We have in 3D

```
h    N
=======
1    8
1/2  27
1/4  64
...
=======
```

□

16.1.2 Discretization error

With the previous considerations, we have now a relationship between h and N that we can use to display the discretization error.

Proposition 16.2. *In the approximate limit it holds:*

$$N \sim \Big(\frac{1}{h}\Big)^d$$

yielding

$$h \sim \frac{1}{\sqrt[d]{N}}$$

These relationships allow us to replace h in error estimates by N.

Proposition 16.3 (Linear and quadratic convergence in 1D)**.** *When we say a scheme has a linear or quadratic convergence in 1D, (i.e., $d = 1$) respectively, we mean:*

$$O(h) = O\Big(\frac{1}{N}\Big)$$

or

$$O(h^2) = O\Big(\frac{1}{N^2}\Big)$$

In a linear scheme, the error will be divided by a factor of 2 when the mesh size h is divided by 2 and having quadratic convergence the error will decrease by a factor of 4.

Proposition 16.4 (Linear and quadratic convergence in 2D)**.** *When we say a scheme has a linear or quadratic convergence in 2D, (i.e., $d = 2$) respectively, we mean:*

$$O(h) = O\Big(\frac{1}{\sqrt{N}}\Big)$$

or

$$O(h^2) = O\Big(\frac{1}{N}\Big)$$

16.1.3 Computationally-obtained convergence order

In order to calculate the convergence order α from numerical results, we make the following derivation. Let $P(k) \to P$ for $k \to 0$ be a converging process and assume that

$$P(k) - \tilde{P} = O(k^\alpha).$$

Here $\tilde{P}$ is either the exact limit P (in case it is known) or some 'good' approximation to it. Let us assume that three numerical solutions are known (this is the minimum number if the limit P is not known). That is

$$P(k), \quad P(k/2), \quad P(k/4).$$

Then, the convergence order can be calculated via the formal approach $P(k) - \tilde{P} = ck^\alpha$ with the following formula:

Proposition 16.5 (Computationally-obtained convergence order)**.** *Given three numerically-obtained values $P(k), P(k/2)$ and $P(k/4)$, the convergence order can be estimated as:*

$$\alpha = \frac{1}{log(2)} log\Big(\big|\frac{P(k) - P(k/2)}{P(k/2) - P(k/4)}\big|\Big). \tag{349}$$

The order α is an estimate and heuristic because we assumed a priori a given order, which strictly speaking we have to proof first.

Proof. We assume:

$$P(k) - P(k/2) = O(k^\alpha),$$
$$P(k/2) - P(k/4) = O((k/2)^\alpha).$$

First, we have

$$P(k/2) - P(k/4) = O((k/2)^\alpha) = \frac{1}{2^\alpha} O(k^\alpha)$$

We simply re-arrange:

$$\begin{aligned} P(k/2) - P(k/4) &= \frac{1}{2^\alpha}\Big(P(k) - P(k/2)\Big) \\ \Rightarrow \quad 2^\alpha &= \frac{P(k) - P(k/2)}{P(k/2) - P(k/4)} \\ \Rightarrow \quad \alpha &= \frac{1}{log(2)} \frac{P(k) - P(k/2)}{P(k/2) - P(k/4)} \end{aligned}$$

□

16.1.4 Temporal order for FE, BE, CN in ODEs

In this section we demonstrate our algorithmic developments in terms of a numerical example. The programming code is written in octave (which is the open-source sister of MATLAB) presented in Section 17.2.6.

16.1.4.1 Problem statement We solve our ODE model problem numerically. Let $a = g - m$ be $a = 0.25$ (test 1) or $a = -0.25$ (test 2) or $a = -10$ (test 3). The IVP is given by:

$$y' = ay, \quad y(t_0 = 2011) = 2.$$

The end time value is $T = 2014$. The tasks are:

- Use the forward Euler (FE), backward Euler (BE), and trapezoidal rule (CN) for the numerical approximation.
- Please observe the accuracy in terms of the discretization error.
- Observe for (stiff) equations with a large negative coefficient $a = -10 \ll 1$ the behavior of the three schemes.

16.1.4.2 Discussion of the results for test 1 $a = 0.25$ In the following, we present our results for the end time value $y(T = 2014)$ for test case 1 ($a = 0.25$) on three mesh levels:

Scheme	#steps(N)	k	y(T)
FE	8	0.37500	4.0961
BE	8	0.37500	4.3959
CN	8	0.37500	4.2363
FE	16	0.18750	4.1624
BE	16	0.18750	4.3115
CN	16	0.18750	4.2346
FE	32	0.093750	4.1975
BE	32	0.093750	4.2720
CN	32	0.093750	4.2341

- In the second column, i.e., $8, 16, 32$, the number of steps (= number of intervals, i.e., so called mesh cells - speaking in PDE terminology) are given. In the column after, the errors are provided.
- In order to compute numerically the convergence order α with the help of formula (349), we work with $k = k_{max} = 0.375$. Then we identify in the above table that $P(k_{max}) = P(0.375) = |y(T) - y_8|, P(k_{max}/2) = P(0.1875) = |y(T) - y_{16}|$ and $P(k_{max}/4) = P(0.09375) = |y(T) - y_{32}|$.
- We monitor that doubling the number of intervals (i.e., halving the step size k) reduces the error in the forward and backward Euler scheme by a factor of 2. This is (almost) linear convergence, which is confirmed by using Formula (349) yielding $\alpha = 1.0921$. The trapezoidal rule is much more accurate (for instance using $N = 8$ the error is 0.2% rather than $13 - 16\%$) and we observe that the error is reduced by a factor of 4. Thus quadratic convergence is detected. Here the 'exact' order on these three mesh levels is $\alpha = 2.0022$.
- A further observation is that the forward Euler scheme is unstable for $N = 16$ and $a = -10$ and has a zig-zag curve, whereas the other two schemes follow the exact solution and the decreasing exp-function. But for sufficiently small step sizes, the forward Euler scheme is also stable which we know from our A-stability calculations. These step sizes can be explicitely determined for this ODE model problem and shown below.

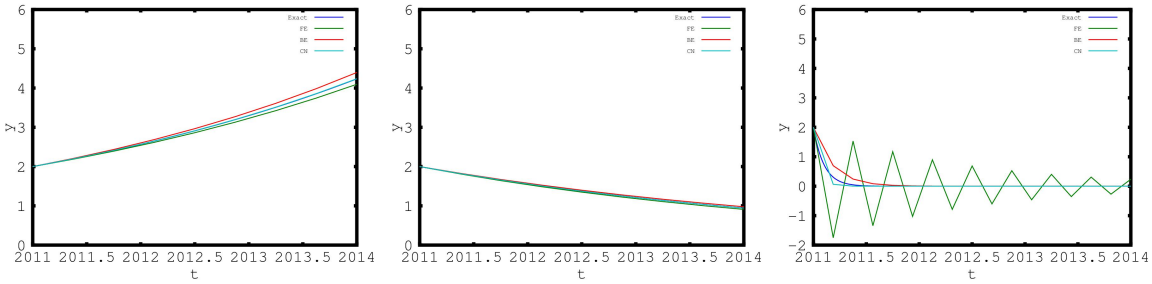


Figure 69: Example 16.1.4.1: on the left, the solution to test 1 is shown. In the middle, test 2 is plotted. On the right, the solution of test 3 with $N = 16$ (number of intervals) is shown. Here, $N = 16$ corresponds to a step size $k = 0.18$ which is slightly below the critical step size for convergence (see Section 16.1.4.3). Thus we observe the instable behavior of the forward Euler method, but also see slow convergence towards the continuous solution.

16.1.4.3 Investigating the instability of the forward Euler method: test 3 $a = -10$ With the help of Example 12.32 let us understand how to choose stable step sizes k for the forward Euler method. The convergence interval reads:

$$|1 + z| \leq 1 \quad \Rightarrow \quad |1 + ak| \leq 1$$

In test 3, $a = -10$, which yields:

$$|1 + z| \leq 1 \quad \Rightarrow \quad |1 - 10k| \leq 1$$

Thus, we need to choose a k that fulfills the previous relation. In this case this, $k < 0.2$ is calculated. This means that for all $k < 0.2$ we should have convergence of the forward Euler method and for $k \geq 0.2$ non-convergence (and in particular no stability!). We perform the following additional tests:

- Test 3a: $N = 10$, yielding $k = 0.3$;
- Test 3b: $N = 15$, yielding $k = 0.2$; exactly the boundary of the stability interval;
- Test 3c: $N = 16$, yielding $k = 0.1875$; from before;
- Test 3d: $N = 20$, yielding $k = 0.15$.

The results of test 3a,3b,3d are provided in Figure 70 and visualize very nicely the theoretically predicted behavior.

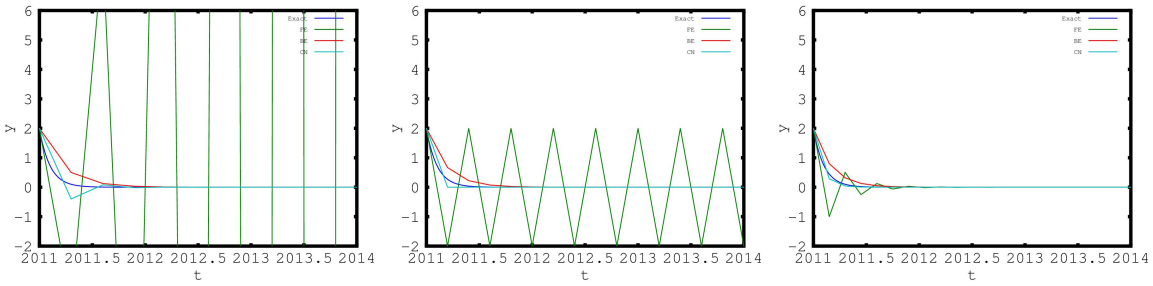


Figure 70: Example 16.1.4.1: tests 3a,3b,3d: Blow-up, constant zig-zag non-convergence, and convergence of the forward Euler method.

Exercise 12. *Consider the ODE-IVP:*

$$y'(t) = ay(t), \quad y(t_0) = 7,$$

where $t_0 = 5$ *and* $T = 11$.

1. *Implement the backward Euler scheme. and set the model parameter to* $a = 2$
2. *Run a second test with* $a = -2$.
3. *Implement the forward Euler scheme. What is the critical step size (Hint: Determine the stability interval).*
4. *Implement the trapezoidal scheme.*
5. *Perform a computational analysis and detect the convergence order.*

Exercise 13. *Implement the predator-prey system and study the solution behavior for different time steps and the convergence order. Details will be given in the lecture.*

Remark 16.6. *Another example where the previous formula is used can be found in Section 8.16.4.*

16.2 Spatial discretization error

We simply use now h and then

$$\alpha = \frac{1}{log(2)} log\Big(\Big|\frac{P(h) - P(h/2)}{P(h/2) - P(h/4)}\Big|\Big) \tag{350}$$

in order to obtain the computational convergence order α. Here,

$$P(h) := u_h, \quad P(h/2) := u_{h/2}, \qquad P(h/4) := u_{h/4},$$

where u is the discrete PDE solution.

16.3 Temporal discretization error

If in time-dependent PDEs, no analytical solution can be constructed, the convergence can again be purely numerically determined. However, we have now influence from spatial and temporal components. Often, the spatial error is more influential than the temporal one. This requires that a numerical reference solution must be computed for sufficiently small h and k. If now 'coarse' temporal solutions are adopted (k coarse!), the spatial mesh must be sufficiently fine, i.e., h small!, because otherwise we see significant influences by the spatial components. Then, we perform the procedure from the previous sections: bisection of k and observing the descrease in the corresponding norm or goal functional. That is to say, observe for a discrete PDE solution u:

$$u_{k,h_{fixed,small}}, \quad u_{k/2,h_{fixed,small}}, \quad u_{k/4,h_{fixed,small}}, \quad \ldots, \quad u_{k_{ref},h_{fixed,small}}$$

With this, we can set:

$$\tilde{P} := u_{k_{ref},h_{fixed,small}}, \quad P(k) = u_{k,h_{fixed,small}}, \quad P(k/2) = u_{k/2,h_{fixed,small}}.$$

Of course, using $P(k/4)$ is also an option. But keep in mind that always h must be sufficiently small and fixed while the time step size is varied.

16.4 Extrapolation to the limit

In case, there is no manufactured solution, the numerical reference value can be improved by extrapolation:

Formulation 16.7. *Let h_1 and h_2 two mesh sizes with $h_2 < h_1$. The functionals of interest are denoted by J_{h_1} and J_{h_2}. Then, the extrapolation formula reads:*

$$J_{extra} = \frac{(h_1 * h_1 * J_{h_2} - h_2 * h_2 * J_{h_1})}{h_1 * h_1 - h_2 * h_2}$$

The value for J_{extra} can then be used as approximation for $\tilde{P}$ from the previous subsection.
A specific example from [114] realized in octave is:

```
% Octave code for the extrapolation to the limit
format long

h1 = 1/2^3 % Mesh size h1
h2 = 1/2^4 % Mesh size h2

% Richter/Wick, Springer, 2017, page 383
p1 = 0.784969295 % Functional value J on h1
p2 = 0.786079344 % Functional value J on h2

% Formula MHB, page 335 (Romberg/Richardson)
% See also Quarteroni, Saleri, Ger. 2014, Springer, extrapolation

pMaHB = p2 + (p2 - p1) / ((h1*h1)/(h2*h2) - 1)

% Formula Richter/Wick for extrapolated value
pRiWi = (h1*h1 * p2 - h2*h2 * p1) / (h1*h1 - h2*h2)
```

16.5 Iteration error

Iterative schemes are used to approximate the discrete solution u_h. This has a priori nothing to do with the discretization error. The main interest is how fast can we get a good approximation of the discrete solution u_h. One example can be found for solving implicit methods for ODEs in which Newton's method is used to compute the discrete solutions of the backward Euler scheme.

To speak about convergence, we compare two subsequent iterations:

Proposition 16.8. *Let us assume that we have an iterative scheme to compute a root z. The iteration converges with order p when*

$$\|x_k - z\| \leq c \, \|x_{k-1} - z\|^p, \quad k = 1, 2, 3, \ldots$$

with $p \geq 1$ and $c = const$. In more detail:

- *Linear convergence: $c \in (0, 1)$ and $p = 1$;*
- *Superlinear convergence: $c := c_k \to 0$, $(k \to \infty)$ and $p = 1$;*
- *Quadratic convergence $c \in \mathbb{R}$ and $p = 2$.*

Cupic and higher convergence are defined as quadratic convergence with the respectice p.

Remark 16.9 (Other characterizations of superlinear and quadratic convergence)**.** *Other (but equivalent) formulations for superlinear and quadratic convergence, respectively, in the case $z \neq x_k$ for all k, are:*

$$\lim_{k \to \infty} \frac{\|x_k - z\|}{\|x_{k-1} - z\|} = 0,$$
$$\limsup_{k \to \infty} \frac{\|x_k - z\|}{\|x_{k-1} - z\|^2} < \infty.$$

Corollary 16.10 (Rule of thumb)**.** *A rule of thumb for quadratic convergence is: the number of correct digits doubles at each step. For instance, a Newton scheme to compute $f(x) = x - \sqrt{2} = 0$ yields the following results:*

```
Iter x            f(x)
==============================
0    3.000000e+00 7.000000e+00
1    1.833333e+00 1.361111e+00
2    1.462121e+00 1.377984e-01
3    1.414998e+00 2.220557e-03
4    1.414214e+00 6.156754e-07
5    1.414214e+00 4.751755e-14
==============================
```

The programming code to this example can be found in [122].

17 Software and programming codes

There are many software packages around for simulating ODEs and PDEs. A google search with the search term 'List of finite element software packages'. A search in October 2020 yielded 44 entries listed on the Wikipedia page https://en.wikipedia.org/wiki/List_of_finite_element_software_packages

17.1 Research software (C++)

Our own sofwares and implementations are based on:

- deal.II www.dealii.org : Poisson's problem (basic version): step-3 https://www.dealii.org/current/doxygen/deal.II/step_3.html
- DOpElib www.dopelib.org : Poisson's problem (basic version) : PDE/StatPDE/Example 4 (2D) and PDE/StatPDE/Example 6 (3D)

17.2 Programming codes

17.2.1 deal.II (C++) step-3 in a 1D version: clothesline / Poisson problem

```
// Thomas Wick
// Nov 2019
// Slight adaptation of deal.II www.dealii.org step-3
#include <deal.II/grid/tria.h>
#include <deal.II/dofs/dof_handler.h>
#include <deal.II/grid/grid_generator.h>
#include <deal.II/grid/tria_accessor.h>
#include <deal.II/grid/tria_iterator.h>
#include <deal.II/dofs/dof_accessor.h>
#include <deal.II/fe/fe_q.h>
#include <deal.II/dofs/dof_tools.h>
#include <deal.II/fe/fe_values.h>
#include <deal.II/base/quadrature_lib.h>
#include <deal.II/base/function.h>
#include <deal.II/numerics/vector_tools.h>
#include <deal.II/numerics/matrix_tools.h>
#include <deal.II/lac/vector.h>
#include <deal.II/lac/full_matrix.h>
#include <deal.II/lac/sparse_matrix.h>
#include <deal.II/lac/dynamic_sparsity_pattern.h>
#include <deal.II/lac/solver_cg.h>
#include <deal.II/lac/precondition.h>
#include <deal.II/numerics/data_out.h>
#include <fstream>
#include <iostream>

using namespace dealii;

class Step3
{
public:
  Step3 ();
  void run ();

private:
  void make_grid ();
```

```
  void setup_system ();
  void assemble_system ();
  void solve ();
  void output_results (unsigned int ref_level) const;

  Triangulation<1>     triangulation;
  FE_Q<1>              fe;
  DoFHandler<1>        dof_handler;

  SparsityPattern      sparsity_pattern;
  SparseMatrix<double> system_matrix;

  Vector<double>       solution;
  Vector<double>       system_rhs;
};


Step3::Step3 ()
  :
  fe (2),
  dof_handler (triangulation)
{}

void Step3::make_grid ()
{
  GridGenerator::hyper_cube (triangulation, 0, 1);
  triangulation.refine_global (1);

  std::cout << "Number of active cells: "
            << triangulation.n_active_cells()
            << std::endl;
}

void Step3::setup_system ()
{
  dof_handler.distribute_dofs (fe);
  std::cout << "Number of degrees of freedom: "
            << dof_handler.n_dofs()
            << std::endl;

  DynamicSparsityPattern dsp(dof_handler.n_dofs());
  DoFTools::make_sparsity_pattern (dof_handler, dsp);
  sparsity_pattern.copy_from(dsp);

  system_matrix.reinit (sparsity_pattern);

  solution.reinit (dof_handler.n_dofs());
  system_rhs.reinit (dof_handler.n_dofs());
}

void Step3::assemble_system ()
{
  QGauss<1>  quadrature_formula(2);
  FEValues<1> fe_values (fe, quadrature_formula,
```

```
                        update_values | update_gradients | update_JxW_values);

  const unsigned int   dofs_per_cell = fe.dofs_per_cell;
  const unsigned int   n_q_points    = quadrature_formula.size();

  FullMatrix<double>   cell_matrix (dofs_per_cell, dofs_per_cell);
  Vector<double>       cell_rhs (dofs_per_cell);

  std::vector<types::global_dof_index> local_dof_indices (dofs_per_cell);

  DoFHandler<1>::active_cell_iterator cell = dof_handler.begin_active();
  DoFHandler<1>::active_cell_iterator endc = dof_handler.end();
  for (; cell!=endc; ++cell)
    {
      fe_values.reinit (cell);

      cell_matrix = 0;
      cell_rhs = 0;

      for (unsigned int q_index=0; q_index<n_q_points; ++q_index)
        {
          for (unsigned int i=0; i<dofs_per_cell; ++i)
            for (unsigned int j=0; j<dofs_per_cell; ++j)
              cell_matrix(i,j) += (fe_values.shape_grad (i, q_index) *
                                   fe_values.shape_grad (j, q_index) *
                                   fe_values.JxW (q_index));

          for (unsigned int i=0; i<dofs_per_cell; ++i)
            cell_rhs(i) += (fe_values.shape_value (i, q_index) *
                            (-1.0) *
                            fe_values.JxW (q_index));
        }
      cell->get_dof_indices (local_dof_indices);

      for (unsigned int i=0; i<dofs_per_cell; ++i)
        for (unsigned int j=0; j<dofs_per_cell; ++j)
          system_matrix.add (local_dof_indices[i],
                             local_dof_indices[j],
                             cell_matrix(i,j));

      for (unsigned int i=0; i<dofs_per_cell; ++i)
        system_rhs(local_dof_indices[i]) += cell_rhs(i);
    }


  std::map<types::global_dof_index,double> boundary_values;
  VectorTools::interpolate_boundary_values (dof_handler,
                                            0,
                                            ZeroFunction<1>(),
                                            boundary_values);

VectorTools::interpolate_boundary_values (dof_handler,
                                            1,
                                            ZeroFunction<1>(),
```

```
                                          boundary_values);

 MatrixTools::apply_boundary_values (boundary_values,
                                     system_matrix,
                                     solution,
                                     system_rhs);
}

void Step3::solve ()
{
  SolverControl           solver_control (1000, 1e-12);
  SolverCG<>              solver (solver_control);

  solver.solve (system_matrix, solution, system_rhs,
                PreconditionIdentity());
}

void Step3::output_results (unsigned int ref_level) const
{
  DataOut<1> data_out;
  data_out.attach_dof_handler (dof_handler);
  data_out.add_data_vector (solution, "solution");
  data_out.build_patches (2);

  //std::ofstream output ("solution.gpl");

  std::string filename_basis;
  filename_basis  = "solution_";

  std::ostringstream filename;
  filename << filename_basis
   << Utilities::int_to_string (ref_level, 4)
   << ".gpl";

  std::ofstream output (filename.str().c_str());

  data_out.write_gnuplot (output);
}

void Step3::run ()
{
  make_grid ();

  unsigned int max_ref = 5;
  for (unsigned int level=0; level<max_ref; level++)
    {
      if (level > 0)
triangulation.refine_global(1);

      setup_system ();
      assemble_system ();
      solve ();
      output_results (level);
    }
```

```
}

int main ()
{
  deallog.depth_console (2);

  Step3 laplace_problem;
  laplace_problem.run ();

  return 0;
}
```

17.2.2 Poisson 1D in python

Python is a programming language that currently enjoys a lot of popularity in scientific computing because of artificial intelligence and machine learning. A website to get started and further information is https://www.python.org/about/gettingstarted/.

The programming code is based on python version 2.7. In order to perform scientific computing with python some additional packages might have to be installed first (on some computers they are pre-installed though):

- numpy
- scipy
- matplotlib

The full code to obtain the previous results is:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# Thomas Wick
# Ecole Polytechnique
# MAP 502
# Winter 2016/2017

# Execution of *.py files
# Possiblity 1: in terminal
#  terminal> python3 file.py  # here file.py = poisson.py

# Possiblity 2: executing in an python environment
# such as spyder.

# This program has been tested with python
# version 2.7 on a SUSE linux system version 13.2

# Problem statement
# Numerical solution using finite
# elements of the Poisson problem:
#
# Find u such that
#
#      -d/dx (k d/dx) u = f in (0,1)
#       u(0) = u(1) = 0
#
#      with f=-1
```

```
##########################################################
# Load packages (need to be installed first if
# not yet done - but is not difficult)
import numpy as np
import matplotlib.pyplot as plt # for plot functons
plt.switch_backend('tkAgg')  # necessary for OS SUSE 13.1 version,
# otherwise, the plt.show() function will not display any window


import scipy as sp
from scipy import sparse
from scipy.sparse import linalg

import scipy.linalg # for the linear solution with Cholesky


##########################################################
# Setup of discretization and material parameters
N = 200 # Number of points in the discretization
x = np.linspace(0,1,N) # Initialize nodal points of the mesh

# Spatial discretization parameter (step length) : h_i=x_{i+1}-x_i
hx = np.zeros(len(x))
hx[:len(x)-1]=x[1:]-x[:len(x)-1]

h = max(hx) # Maximal element diameter

# Sanity check
#print h

# Material parameter
k= 1.0 * np.ones(len(x))


##########################################################
# Construct and plot exact solution and right hand side f

Nddl = N-2
xddl = x.copy()
xddl = xddl[1:len(x)-1]

# Initialize and construct exact solution
uexact = np.zeros(len(x))
uexact[1:len(x)-1] = 0.5 * (xddl * xddl - xddl)

# Construct right hand side vector
f = (-1) * np.ones(len(x)-2)


##########################################################
# Build system matrix and right hand side
Kjj = k[:N-2]/hx[:N-2] + k[1:N-1]/hx[1:N-1] # the diagonal of Kh
Kjjp = - k[1:N-2]/hx[1:N-2] # the two off-diagionals of Kh
```

```
# System matrix (also commenly known as stiffness matrix)
# First argument: the entries
# Second argument: in which diagonals shall the entries be sorted
Ah =sp.sparse.diags([Kjjp,Kjj,Kjjp],[-1,0,1],shape=(Nddl,Nddl))

# Right hand side vector
Fh= (hx[:N-2] + hx[1:N-1]) * 0.5 * f


############################################################
# Solve system: Ah uh = Fh => uh = Ah^{-1}Fh
utilde = sp.sparse.linalg.spsolve(Ah,Fh)

# Add boundary conditions to uh
# The first and last entry are known to be zero.
# Therefore, we include them explicitely:
uh = np.zeros(len(uexact))
uh[1:len(x)-1]=utilde


############################################################
# Plot solution
plt.clf()
plt.plot(x,uexact,'r', label='Exact solution $u$')
plt.plot(x,uh,'b+',label='Discrete FE solution $u_h$')
plt.legend(loc='best')
plt.xlim([-0.1,1.1])
plt.ylim([-0.15,0.01])
plt.show()
```

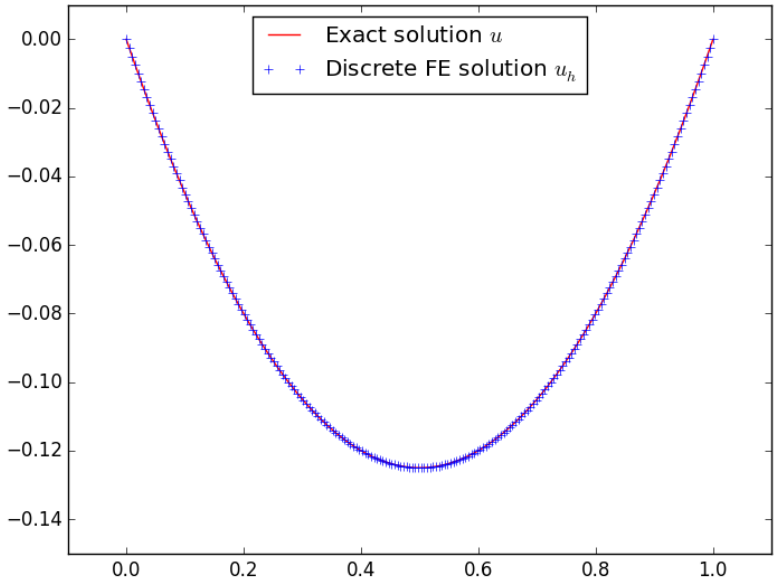


Figure 71: Solution of the 1D Poisson problem.

17.2.3 Poisson 1D in python (short-elegant version) by Roth/Schröder

This code is based on python3. Here are some hints assuming you have python3 installed, but some packages are still missing.

```
# pip  is the installer for python
# pip3 is the installer for python3
# Find out current versions
> pip3 --version
> python3 --version

# Numerical and visualization packages
> pip3 install --user scipy
> pip3 install --user matplotlib

# Graphical user interface (GUI):
> pip3 install --user PyQt5==5.9.2

# Running the program. Here the CODE FROM BELOW has to
# be copied into the file poisson.py
> python3 poisson.py
```

Here is now the code copied from the file poisson.py:

```
import numpy as np
from scipy.sparse import diags
from scipy.sparse.linalg import spsolve
```

```
import matplotlib.pyplot as plt

N = 10
h = 1 / N
x = np.linspace(0, 1, N + 1)
xFine = np.linspace(0, 1, 5 * N)

print(f"There are {N+1} nodal points and the mesh size is {h}.")

# assemble system
systemMatrix = (1 / h) * diags(
    [-1, 2, -1], [-1, 0, 1], shape=(N + 1, N + 1), format="csr"
)
rightHandSide = -h * np.ones(N + 1)

# enforce homogeneous Dirichlet boundary conditions
systemMatrix[0, 0] = 1.0
systemMatrix[0, 1] = 0.0
systemMatrix[N, N] = 1.0
systemMatrix[N, N - 1] = 0.0

rightHandSide[0] = 0.0
rightHandSide[N] = 0.0

# solve linear system
solution = spsolve(systemMatrix, rightHandSide)
analyticSolution = 0.5 * xFine * (xFine - 1)

plt.plot(xFine, analyticSolution, "r", label="Analytic solution $u$")
plt.plot(x, solution, "b+", label="FEM solution $u_h$")
plt.legend(loc="best")
plt.show()
```

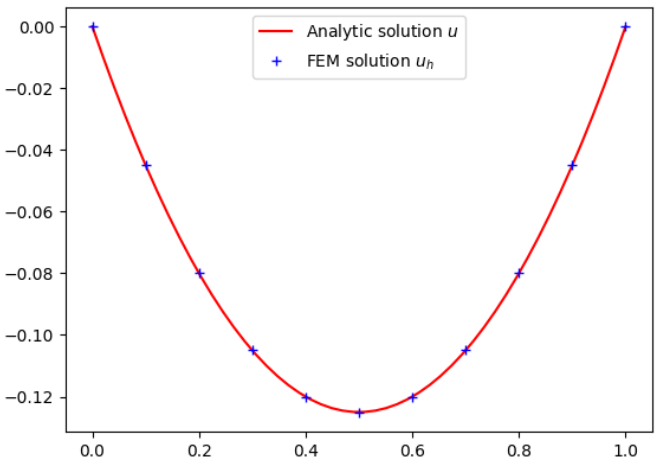


Figure 72: Solution of the 1D Poisson problem.

17.2.4 Poisson 2D in python with assembling and numerical quadrature by Roth/Schröder

In this code, we show also in detail the assembling with test functions and numerical quadrature. The results are analyzed with a figure and also the evaluation of a goal functional.

```
import numpy as np
from scipy.sparse import dok_matrix
from scipy.sparse.linalg import spsolve
import matplotlib.pyplot as plt

# right-hand side
f = -1.0

# quadrature points and weights
quad = simpson = [(0.0, 1 / 6), (0.5, 4 / 6), (1.0, 1 / 6)]  # simpson rule

# basis functions in 1D
hat0 = {"eval": lambda x: x, "nabla": lambda x: 1}
hat1 = {"eval": lambda x: 1 - x, "nabla": lambda x: -1}
hatFunction = [hat0, hat1]

# basis functions in 2D = tensor product of 1D basis functions
class Phi2D:
    def __init__(self, x0, y0):
        self.xBasis = hatFunction[x0]
        self.yBasis = hatFunction[y0]

    # function evaluation
    def eval(self, x, y):
        return self.xBasis["eval"](x) * self.yBasis["eval"](y)

    # derivative
    def nabla(self, x, y):
        return np.array(
            [
                self.xBasis["nabla"](x) * self.yBasis["eval"](y),
                self.xBasis["eval"](x) * self.yBasis["nabla"](y),
            ]
        )


# linear basis functions in 2D
basisFunction = {0: Phi2D(0, 0), 1: Phi2D(1, 0), 2: Phi2D(0, 1), 3: Phi2D(1, 1)}


class Cell:
    def __init__(self, origin, right, up, upRight):
        self.dofs = [origin, right, up, upRight]


class DoF:
    def __init__(self, x, y, ind=-1):
        self.x, self.y = x, y
        self.ind = ind
```

```
class Grid:
    def __init__(self, xMin=0.0, xMax=1.0, yMin=0.0, yMax=1.0, stepSize=0.5):
        self.xMin, self.xMax = xMin, xMax
        self.yMin, self.yMax = yMin, yMax
        self.h = stepSize
        self.dofs, self.cells = [], []

    def decompose(self):
        # create mesh and transform it into a list of x and y coordinates
        xRange = np.arange(self.xMin, self.xMax, self.h)
        xRange = np.append(xRange, [self.xMax])
        self.xNum = len(xRange)
        yRange = np.arange(self.yMin, self.yMax, self.h)
        yRange = np.append(yRange, [self.yMax])
        self.yNum = len(yRange)
        self.xCoord, self.yCoord = np.meshgrid(xRange, yRange)
        xList, yList = self.xCoord.ravel(), self.yCoord.ravel()

        # create DoFs
        for i, (x, y) in enumerate(zip(xList, yList)):
            self.dofs.append(DoF(x, y, ind=i))

        # create cells
        for i, dof in enumerate(self.dofs):
            if dof.x != self.xMax and dof.y != self.yMax:
                self.cells.append(
                    Cell(
                        dof,
                        self.dofs[i + 1],
                        self.dofs[i + self.xNum],
                        self.dofs[i + self.xNum + 1],
                    )
                )

    def assembleSystem(self):
        # system matrix
        A = dok_matrix((len(self.dofs), len(self.dofs)), dtype=np.float32)
        # system right hand side
        F = np.zeros(len(self.dofs), dtype=np.float32)

        for cell in self.cells:
            for x, y, quadWeight in
                [ (x, y, wX * wY) for (x, wX) in quad for (y, wY) in quad ]:
                for j, dof_j in enumerate(cell.dofs):
                    # assemble rhs
                    F[dof_j.ind] += (
                        quadWeight * f * basisFunction[j].eval(x, y) * self.h ** 2
                    )
                    for i, dof_i in enumerate(cell.dofs):
                        # assemble matrix
                        A[dof_i.ind, dof_j.ind] += quadWeight
                            * basisFunction[i].nabla(x, y).dot(
                              basisFunction[j].nabla(x, y)
```

```
                        )

        # apply homogeneous Dirichlet boundary conditions
        for dof in self.dofs:
            if dof.x in [self.xMin, self.xMax] or dof.y in [self.yMin, self.yMax]:
                _, nonZeroColumns = A[dof.ind, :].nonzero()
                for j in nonZeroColumns:
                    A[dof.ind, j] = 0.0
                A[dof.ind, dof.ind] = 1.0
                F[dof.ind] = 0.0

        return A.tocsr(), F

    def plotSolution(self, solution):
        # 2D contour plot
        plt.contourf(
            self.xCoord,
            self.yCoord,
            solution.reshape(self.yNum, self.xNum),
            20,
            cmap="viridis",
        )
        plt.colorbar()
        plt.show()

    def printSolutionAtCenter(self,solution):
        #prints the solution at center of the domain
        xCenter = (self.xMax + self.xMin)/2
        yCenter = (self.yMax + self.yMin)/2
        for dof in self.dofs:
            if abs(dof.x - xCenter) < 1e-10 and abs(dof.y - yCenter) < 1e-10:
                print(f'The Solution at the center x={xCenter},y={yCenter}
                        has the value {solution[dof.ind]}.')
                break
        else:
            print(f'The center point x={xCenter},y={yCenter} is not on the grid.')



if __name__ == "__main__":
    grid = Grid(stepSize=0.1)
    grid.decompose()
    A, F = grid.assembleSystem()
    solution = spsolve(A, F)  # U = A^{-1}F
    grid.printSolutionAtCenter(solution)
    grid.plotSolution(solution)
```

Goal functional evaluation:

```
The Solution at the center x=0.5,y=0.5 has the value -0.07426007837057114.
```

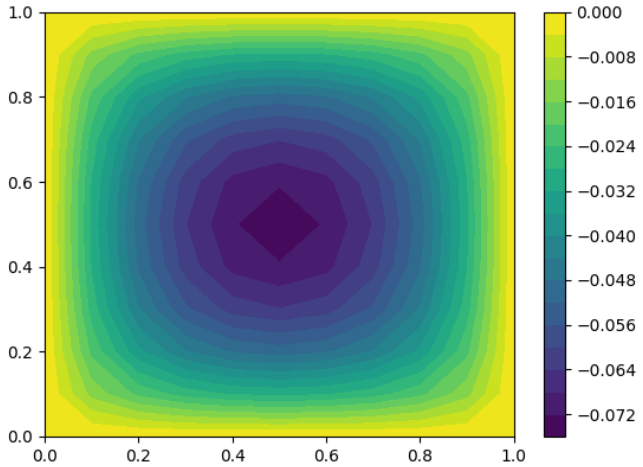


Figure 73: Solution of the 2D Poisson problem.

17.2.5 Newton's method for solving nonlinear problems: octave/MATLAB and C++

We present three variants:

1. Root-finding with a standard Newton method for a real-valued function (octave)
2. Root-finding with a defect-correction Newton method for a real-valued function (octave)
3. Root-finding (solving!) a nonlinear PDE (C++ with deal.II)

17.2.5.1 Code for a straightforward implementation of Newton's method The programming code for standard Newton's method is as follows:

```
% Script file: step_2.m
% Author: Thomas Wick, CMAP, Ecole Polytechnique, 2016
% We seek a root of a nonlinear problem

format long

% Define nonlinear function of which
% we want to find the root
f = @(x) x^2 - 2;

% Define derivative of the nonlinear function
df = @(x) 2*x;

% We need an initial guess
x = 3;

% Define function
y = f(x);

% Set iteration counter
it_counter = 0;
printf('%i %e %e\n',it_counter,x,y);
```

```
% Set stopping tolerance
TOL = 1e-10;

% Perform Newton loop
while abs(y) > TOL
   dy = df(x);
   x = x - y/dy;
   y = f(x);
   it_counter = it_counter + 1;
   printf('%i %e %e\n',it_counter,x,y);
end
```

17.2.5.2 Code for a Newton defect-correction scheme including line search As second implementation we present Newton's method as defect correction scheme as introduced in Definition 13.20. Additionally we introduce a line search parameter $\omega \in [0, 1]$. If the initial Newton guess is not good enough we can enlarge the convergence radius of Newton's method by choosing $\omega < 1$. The octave code reads:

```
% Script file: step_3.m
% Author: Thomas Wick, CMAP, Ecole Polytechnique, 2016
% We seek a root of a nonlinear problem

format long

% Define nonlinear function of which
% we want to find the root
f = @(x) x^2 - 2;

% Define derivative of the nonlinear function
df = @(x) 2*x;

% We need an initial guess
x = 3;

% Define function
y = f(x);

% Set iteration counter
it_counter = 0;
printf('%i %e %e\n',it_counter,x,y);

% Set stopping tolerance
TOL = 1e-10;

% Line search parameter
% omega \in [0,1]
% where omega = 1 corresponds to full Newton
omega = 1.0;

% Newtons method as defect correction
% and line search parameter omega
while abs(y) > TOL
   dy = df(x);
   % Defect step
   dx = - y/dy;
```

```
   % Correction step
  x = x + omega * dx;
   y = f(x);
   it_counter = it_counter + 1;
   printf('%i %e %e\n',it_counter,x,y);
end
```

17.2.5.3 Pseudo C++ code of a Newton implementation with line search An example of a pseudo C++ code demonstrating the algorithm from Section 13.10.2 was implemented in [137]; see https://media.archnumsoft.org/10305/doc/step-fsi_8cc_source.html and https://media.archnumsoft.org/10305/

This piece of code has been the work horse of the author's own research since the year 2011. In [73], see https://github.com/tjhei/cracks this code was the basis for a parallel version.

```
newton_iteration ()
{
  const double lower_bound_newton_residual = 1.0e-10;
  const unsigned int max_no_newton_steps  = 20;

  // Decision whether the system matrix should be build at each Newton step
  const double nonlinear_theta = 0.1;

  // Line search parameters
  unsigned int line_search_step;
  const unsigned int  max_no_line_search_steps = 10;
  const double line_search_damping = 0.6;
  double new_newton_residual;

  // Application of nonhomogeneous Dirichlet boundary conditions to the variational equations:
  set_nonhomo_Dirichlet_bc ();
  // Evaluate the right hand side residual
  assemble_system_rhs();

  double newton_residual = system_rhs.linfty_norm();
  double old_newton_residual = newton_residual;
  unsigned int newton_step = 1;

  if (newton_residual < lower_bound_newton_residual)
      std::cout << '\t' << std::scientific << newton_residuum << std::endl;

  while (newton_residual > lower_bound_newton_residual && newton_step < max_no_newton_steps)
    {
      old_newton_residual = newton_residual;

      assemble_system_rhs();
      newton_residual = system_rhs.linfty_norm();

      if (newton_residuum < lower_bound_newton_residuum)
        {
          std::cout << '\t'
                    << std::scientific
                    << newton_residual << std::endl;
          break;
        }

      // Simplified Newton steps
      if (newton_residual/old_newton_residual > nonlinear_theta)
        assemble_system_matrix ();

      // Solve linear equation system Ax = b
      solve ();

      line_search_step = 0;
      for ( ; line_search_step < max_no_line_search_steps; ++line_search_step)
         {
          solution += newton_update;
```

```
          assemble_system_rhs ();
          new_newton_residual = system_rhs.linfty_norm();

          if (new_newton_residual < newton_residual)
            break;
          else
            solution -= newton_update;

          newton_update *= line_search_damping;
         }

      // Output to the terminal for the user
      std::cout << std::setprecision(5) <<newton_step << '\t' << std::scientific << newton_residual << '\t'
                << std::scientific << newton_residual/old_newton_residual  <<'\t' ;
      if (newton_residual/old_newton_residual > nonlinear_theta)
        std::cout << "r" << '\t' ;
      else
        std::cout << " " << '\t' ;
      std::cout << line_search_step  << '\t' << std::endl;

      // Goto next newton iteration, increment j->j+1
      newton_step++;
    }
}
```

17.2.6 Time-stepping schemes for ODEs in Octave

In this section, we present the code for Example 16.1.4. Despite that this is only an ODE, it is useful also for PDE because very often time-dependent PDEs are solved in the temporal variable with finite difference schemes as presented here.

The package MATLAB is widely used for scientific computations. MATLAB is a trademark of The-MathsWorks Inc., 24 Prime Park Way, Natick, MA 01760. The website is www.mathworks.com. The open source sister language with similar commands is Octave (in detail GNU Octave) with the website www.octave.org. We use the latter one in these notes. For both languages, an excellent introduction can be found in the textbook by Quarteroni et al. [98].

We first define the three numerical schemes. These are defined in terms of octave-functions and which must be in the same directory where you later call these functions and run octave. Thus, the content of the next three functions is written into files with the names:

```
euler_forward_2.m  euler_backward_model_problem.m   trapezoidal_model_problem.m
```

Inside these m-files we have the following statements. Forward Euler:

```
function [x,y] = euler_forward_2(f,xinit,yinit,xfinal,n)
% Forward Euler scheme
% Author: Thomas Wick, CMAP, Ecole Polytechnique, 2016

h = (xfinal - xinit)/n;

x = [xinit zeros(1,n)];
y = [yinit zeros(1,n)];

for k = 1:n
  x(k+1) = x(k)+h;
          y(k+1) = y(k) + h*f(x(k),y(k));
end
end
```

Backward Euler (here the right hand side is also unknown). For simplicity we explicitly manipulate the scheme for this specific ODE. In the general case, the right hand side is solved in terms of a nonlinear solver.

```
function [x,y] = euler_backward_model_problem(xinit,yinit,xfinal,n,a)
% Backward Euler scheme
% Author: Thomas Wick, CMAP, Ecole Polytechnique, 2016

% Calculuate step size
h = (xfinal - xinit)/n;

% Initialize x and y as column vectors
x = [xinit zeros(1,n)];
y = [yinit zeros(1,n)];

% Implement method
for k = 1:n
  x(k+1) = x(k)+h;
          y(k+1) = 1./(1.0-h.*a) .*y(k);
end
end
```

And finally the trapezoidal rule (Crank-Nicolson):

```
function [x,y] = trapezoidal_model_problem(xinit,yinit,xfinal,n,a)
% Trapezoidal / Crank-Nicolson
% Author: Thomas Wick, CMAP, Ecole Polytechnique, 2016

% Calculuate step size
h = (xfinal - xinit)/n;

% Initialize x and y as column vectors
x = [xinit zeros(1,n)];
y = [yinit zeros(1,n)];

% Implement method
for k = 1:n
  x(k+1) = x(k)+h;
          y(k+1) = 1./(1 - 0.5.*h.*a) .*(1 + 0.5.*h.*a).*y(k);
end
end
```

In order to have a re-usable code in which we can easily change further aspects, we do not work in the command line but define an octave-script; here with the name

```
step_1.m
```

that contains initialization and function calls, visualization, and error estimation:

```
% Script file: step_1.m
% Author: Thomas Wick, CMAP, Ecole Polytechnique, 2016

% We solve three test scenarios
% Test 1: coeff_a = 0.25 as in the lecture notes.
% Test 2: coeff_a = -0.1
% Test 3: coeff_a = -10.0 to demonstrate
%    numerical instabilities using explizit schemes

% Define mathematical model, i.e., ODE
% coeff_a = g - m
coeff_a = 0.25;
```

```
f=@(t,y) coeff_a.*y;

% Initialization
t0 = 2011;
tN   = 2014;
num_steps = 16;
y0 = 2;

% Implement exact solution, which is
% available in this example
g=@(t) y0 .* exp(coeff_a .* (t-t0));
te=[2011:0.01:2014];
ye=[g(te)];

% Call forward and backward Euler methods
[t1,y1] = euler_forward_2(f,t0,y0,tN,num_steps);
[t2,y2] = euler_backward_model_problem(t0,y0,tN,num_steps,coeff_a);
[t3,y3] = trapezoidal_model_problem(t0,y0,tN,num_steps,coeff_a);

% Plot solution
plot (te,ye,t1,y1,t2,y2,t3,y3)
xlabel('t')
ylabel('y')
  legend('Exact','FE','BE','CN')
axis([2011 2014 0 6])

% Estimate relative discretization error
errorFE=['FE err.: ' num2str((ye(end)-y1(end))/ye(end))]
errorBE=['BE err.: ' num2str((ye(end)-y2(end))/ye(end))]
errorCN=['CN err.: ' num2str((ye(end)-y3(end))/ye(end))]

% Estimate absolute discretization error
errorABS_FE=['FE err.: ' num2str((ye(end)-y1(end)))]
errorABS_BE=['BE err.: ' num2str((ye(end)-y2(end)))]
errorABS_CN=['CN err.: ' num2str((ye(end)-y3(end)))]
```

Exercise 14. *Implement the backward Euler scheme and trapezoidal rule in a more general way that avoids the explicit manipulation with the right hand side $f(t, y) = ay(t)$. Hint: A possibility is (assuming that the function f is smooth enough) to formulate the above schemes as fixed-point equations and then either use a fixed-point method or Newton's (see Section 17.2.6.1) method to solve them. Apply these ideas to Example 12.*

17.2.6.1 Background: Solving initial-valued problems with implicit schemes In Section 12.3.7.2, we introduced implicit schemes to solve ODEs. In specific cases, one can explicitely arrange the terms in such a way that an implicit schemes is obtained. This was the procedure for the ODE model problem in Section 17.2.6. However, such an explicit representation is not always possible. Rather we have to solve a nonlinear problem inside the time-stepping process. This nonlinear problem will be formulated in terms of a Newton scheme.

We recall:

$$y'(t) = f(t, y(t))$$

which reads after backward Euler discretization:

$$\frac{y_n - y_{n-1}}{k_n} = f(t_n, y_n(t_n)).$$

After re-arranging some terms we obtain the root-finding problem:

$$g(y_n) := y_n - y_{n-1} - k_n f(t_n, y_n(t_n)) = 0.$$

Taylor expansion yields (where the Newton iteration index is denoted by 'l'):

$$0 = g(y_n^l) + g'(y_n^l)(y_n^{l+1} - y_n^l) + o((y_n^{l+1} - y_n^l)^2),$$

where the Newton matrix (the Jacobian) is given by

$$g'(y_n^l) := I - k_n f_y'(t_n, y_n^l).$$

The derivative f_y' is with respect to the unknown y_n.

In defect correction notation, we have

Algorithm 17.1. *Given a starting value, e.g., y_n^0 we have for $l = 1, 2, 3, \ldots$:*

$$g'(y_n^l)\delta y = -g(y_n^l), \tag{351}$$

$$y_n^{l+1} = y_n^l + \delta y \tag{352}$$

Stop if $\|g(y_n^{l+1})\| < TOL_N$, with e.g., $TOL_N = 10^{-8}$, set $y_n := y_n^{l+1}$.

The difficulty here is to understand the mechanism of the two different indices n and l. As before, we want to compute for a new time point t^{n+1} a solution y_{n+1}. However, in most cases y_{n+1} cannot be constructed explicitely as done for example in Section 4.1.2. Rather we need to approximate y_{n+1} by another numerical scheme. Here Newton's method is one efficient possibility, in which we start with a rough approximation for y_{n+1}, which is denoted as initial Newton guess $y_{n+1}^0 \approx y_{n+1}$. In order to improve this initial guess, we start a Newton iteration labeled by l and (hope!) to find better approximations

$$y_{n+1}^1, y_{n+1}^2, \ldots$$

for the sought value y_{n+1}. Mathematically speaking, we hope that

$$|y_{n+1}^l - y_{n+1}| \to 0, \quad \text{for } l \to \infty.$$

To make this procedure clear:

- we are interested in the true value $y(t^{n+1})$, which is however in most cases not possible to be computed in an explicit way.
- Thus, we use a finite difference scheme (e.g., backward Euler) to approximate $y(t^{n+1})$ by y_{n+1}.
- However, in many cases this finite difference scheme (precisely speaking: implicit schemes) is still not enough to obtain directly y_{n+1}. Therefore, we need a second scheme (here Newton) to approximate y_{n+1} by y_{n+1}^l.

Remark 17.2. *To avoid two numerical schemes is one reason, why explicit schemes are nevertheless very popular in practice. Here the second scheme can be omitted.*

Remark 17.3. *On the other hand, this example is characteristic for many numerical algorithms: very often, we have several algorithms that interact with each other. One important question is how to order these algorithms such that the overall numerical scheme is robust and efficient.*

17.2.7 deal.II (C++) PU-DWR implementation used in Section 9

In this section, we show the key code snippet for the PU-DWR implementation. The full code can be found here https://thomaswick.org/Codes/step_nse_Nov_9_2021.cc

```
////////////////////////////////////////////////////////////////////////////////////////
// PU-DWR Error estimator


// Implementing a weak form of DWR localization using a partition-of-unity
```

```
// as it has been proposed in
//
// T. Richter, T. Wick;
// Variational Localizations of the Dual-Weighted Residual Estimator,
// Journal of Computational and Applied Mathematics,
// Vol. 279 (2015), pp. 192-208
//
// Some routines have been taken from step-14 in deal.II
template<int dim>
double NSE_PU_DWR_Problem<dim>::compute_error_indicators_a_la_PU_DWR (const unsigned int refinement_cy
{

  // First, we re-initialize the error indicator vector that
  // has the length of the space dimension of the PU-FE.
  // Therein we store the local errors at all degrees of freedom.
  // This is in contrast to usual procedures, where the error
  // in general is stored cell-wise.
  dof_handler_pou.distribute_dofs (fe_pou);
  error_indicators.reinit (dof_handler_pou.n_dofs());


  // Block 1 (building the dual weights):
  // In the following the very specific
  // part (z-I_hz) of DWR is implemented.
  // This part is the same for classical error estimation
  // and PU error estimation.
  std::vector<unsigned int> block_component (3,0);
  block_component[dim] = 1;


  DoFRenumbering::component_wise (dof_handler_adjoint, block_component);

  // Implement the interpolation operator
  // (z-z_h)=(z-I_hz)
  ConstraintMatrix dual_hanging_node_constraints;
  DoFTools::make_hanging_node_constraints (dof_handler_adjoint,
   dual_hanging_node_constraints);
  dual_hanging_node_constraints.close();

  ConstraintMatrix primal_hanging_node_constraints;
  DoFTools::make_hanging_node_constraints (dof_handler_primal,
   primal_hanging_node_constraints);
  primal_hanging_node_constraints.close();


  // TODO double-check (3), 0 , 1, 2
  // Construct a local primal solution that
  // has the length of the adjoint vector
  std::vector<unsigned int> dofs_per_block (2);
  DoFTools::count_dofs_per_block (dof_handler_adjoint,
  dofs_per_block, block_component);
  const unsigned int n_v = dofs_per_block[0];
  const unsigned int n_p = dofs_per_block[1];
```

```
  BlockVector<double> solution_primal_of_adjoint_length;
  solution_primal_of_adjoint_length.reinit(2);
  solution_primal_of_adjoint_length.block(0).reinit(n_v);
  solution_primal_of_adjoint_length.block(1).reinit(n_p);
  solution_primal_of_adjoint_length.collect_sizes ();

  // Main function 1: Interpolate cell-wise the
  // primal solution into the dual FE space.
  // This rescaled primal solution is called
  //   ** solution_primal_of_adjoint_length **
  FETools::interpolate (dof_handler_primal,
solution_primal,
dof_handler_adjoint,
dual_hanging_node_constraints,
solution_primal_of_adjoint_length);



  // Local vectors of dual weights obtained
  // from the adjoint solution
  BlockVector<double> dual_weights;
  dual_weights.reinit(2);
  dual_weights.block(0).reinit(n_v);
  dual_weights.block(1).reinit(n_p);
  dual_weights.collect_sizes ();

  // Main function 2: Execute (z-I_hz) (in the dual space),
  // yielding the adjoint weights for error estimation.
  FETools::interpolation_difference (dof_handler_adjoint,
     dual_hanging_node_constraints,
     solution_adjoint,
     dof_handler_primal,
     primal_hanging_node_constraints,
     dual_weights);

  // end Block 1


  // Block 2 (evaluating the PU-DWR):
  // The following function has a loop inside that
  // goes over all cells to collect the error contributions,
  // and is the 'heart' of the DWR method. Therein
  // the specific equation of the error estimator is implemented.


  // Info: must be sufficiently high for adjoint evaluations
  QGauss<dim>   quadrature_formula(5);

  FEValues<dim> fe_values_pou (fe_pou, quadrature_formula,
       update_values    |
       update_quadrature_points  |
       update_JxW_values |
       update_gradients);
```

```
FEValues<dim> fe_values_adjoint (fe_adjoint, quadrature_formula,
 update_values    |
 update_quadrature_points  |
 update_JxW_values |
 update_gradients);


const unsigned int   dofs_per_cell = fe_values_pou.dofs_per_cell;
const unsigned int   n_q_points    = fe_values_pou.n_quadrature_points;

Vector<double>  local_err_ind (dofs_per_cell);

std::vector<unsigned int> local_dof_indices (dofs_per_cell);



const FEValuesExtractors::Vector velocities (0); // 0
 const FEValuesExtractors::Scalar pressure (dim); // 2

 const FEValuesExtractors::Scalar pou_extract (0);


 std::vector<Vector<double> > primal_cell_values (n_q_points,
  Vector<double>(dim+1));

 std::vector<std::vector<Tensor<1,dim> > > primal_cell_gradients (n_q_points,
  std::vector<Tensor<1,dim> > (dim+1));

 std::vector<Vector<double> > dual_weights_values(n_q_points,
   Vector<double>(dim+1));

 std::vector<std::vector<Tensor<1,dim> > > dual_weights_gradients (n_q_points,
   std::vector<Tensor<1,dim> > (dim+1));

 typename DoFHandler<dim>::active_cell_iterator
   cell = dof_handler_pou.begin_active(),
   endc = dof_handler_pou.end();

 typename DoFHandler<dim>::active_cell_iterator
   cell_adjoint = dof_handler_adjoint.begin_active();

 for ( ; cell!=endc; ++cell, ++cell_adjoint)
   {
     fe_values_pou.reinit (cell);
     fe_values_adjoint.reinit (cell_adjoint);

     local_err_ind = 0;


     // primal solution (cell residuals)
     // But we use the adjoint FE since we previously enlarged the
     // primal solution to the length of the adjoint vector.
```

```
      fe_values_adjoint.get_function_values (solution_primal_of_adjoint_length,
     primal_cell_values);

      fe_values_adjoint.get_function_gradients (solution_primal_of_adjoint_length,
primal_cell_gradients);

      // adjoint weights
      fe_values_adjoint.get_function_values (dual_weights,
     dual_weights_values);

      fe_values_adjoint.get_function_gradients (dual_weights,
dual_weights_gradients);




     // Gather local error indicators while running
     // of the degrees of freedom of the partition of unity
     // and corresponding quadrature points.
      for (unsigned int q=0; q<n_q_points; ++q)
{
  // Right hand side
 Tensor<1,dim> fluid_force;
 fluid_force.clear();
 fluid_force[0] = density_fluid * force_fluid_x;
 fluid_force[1] = density_fluid * force_fluid_y;

 // Primal cell values
 Tensor<1,2> v;
 v.clear();
 v[0] = primal_cell_values[q](0);
 v[1] = primal_cell_values[q](1);

 Tensor<2,dim> grad_v;
 grad_v[0][0] = primal_cell_gradients[q][0][0];
 grad_v[0][1] = primal_cell_gradients[q][0][1];
 grad_v[1][0] = primal_cell_gradients[q][1][0];
 grad_v[1][1] = primal_cell_gradients[q][1][1];

 Tensor<2,dim> pI;
 pI[0][0] = primal_cell_values[q](2);
 pI[1][1] = primal_cell_values[q](2);

 // Adjoint weights
 Tensor<1,dim> dw_v;
 dw_v[0] = dual_weights_values[q](0);
 dw_v[1] = dual_weights_values[q](1);

 double dw_p = dual_weights_values[q](2);

 Tensor<2,dim> grad_dw_v;
 grad_dw_v[0][0] = dual_weights_gradients[q][0][0];
 grad_dw_v[0][1] = dual_weights_gradients[q][0][1];
 grad_dw_v[1][0] = dual_weights_gradients[q][1][0];
```

```
  grad_dw_v[1][1] = dual_weights_gradients[q][1][1];



  Tensor<2,dim> stress_term = - pI + density_fluid * viscosity * (grad_v + transpose(grad_v));

  double divergence_v = grad_v[0][0] + grad_v[1][1];

  Tensor<1,2> convection_fluid = density_fluid * grad_v * v;

  // Run over all PU degrees of freedom per cell (namely 4 DoFs for Q1 FE-PU)
  for (unsigned int i=0; i<dofs_per_cell; ++i)
    {

      Tensor<2,dim> grad_phi_psi;
      grad_phi_psi[0][0] = fe_values_pou[pou_extract].value(i,q) * grad_dw_v[0][0] + dw_v[0] * fe_valu
      grad_phi_psi[0][1] = fe_values_pou[pou_extract].value(i,q) * grad_dw_v[0][1] + dw_v[0] * fe_valu
      grad_phi_psi[1][0] = fe_values_pou[pou_extract].value(i,q) * grad_dw_v[1][0] + dw_v[1] * fe_valu
      grad_phi_psi[1][1] = fe_values_pou[pou_extract].value(i,q) * grad_dw_v[1][1] + dw_v[1] * fe_valu



      // Implement the error estimator
      // J(u) - J(u_h) \approx \eta := (f,...) - (\nabla u, ...)
      //
      // First part: (f,...)
      local_err_ind(i) += (fluid_force  * dw_v * fe_values_pou[pou_extract].value(i,q)
   ) * fe_values_pou.JxW (q);

      // Second part: - (\nabla u, ...)
      local_err_ind(i) -= (scalar_product(stress_term, grad_phi_psi)
   + convection_fluid * dw_v * fe_values_pou[pou_extract].value(i,q)
   +  divergence_v * dw_p * fe_values_pou[pou_extract].value(i,q)
   ) * fe_values_pou.JxW (q);


    }

} // end q_points


       // Write all error contributions
       // in their respective places in the global error vector.
       cell->get_dof_indices (local_dof_indices);
       for (unsigned int i=0; i<dofs_per_cell; ++i)
error_indicators(local_dof_indices[i]) += local_err_ind(i);


     } // end cell loop for PU FE elements


  // Finally, we eliminate and distribute hanging nodes in the error estimator
  ConstraintMatrix dual_hanging_node_constraints_pou;
  DoFTools::make_hanging_node_constraints (dof_handler_pou,
```

```
 dual_hanging_node_constraints_pou);
dual_hanging_node_constraints_pou.close();

// Distributing the hanging nodes
dual_hanging_node_constraints_pou.condense(error_indicators);

// Averaging (making the 'solution' continuous)
dual_hanging_node_constraints_pou.distribute(error_indicators);

// end Block 2


// Block 3 (data and terminal print out)
DataOut<dim> data_out;
data_out.attach_dof_handler (dof_handler_pou);
data_out.add_data_vector (error_indicators, "error_ind");
data_out.build_patches ();

std::ostringstream filename;
filename << "solution_error_indicators_"
 << refinement_cycle
 << ".vtk"
 << std::ends;

std::ofstream out (filename.str().c_str());
data_out.write_vtk (out);


// Print out on terminal
std::cout << "-------------------" << std::endl;
std::cout << std::setiosflags(std::ios::scientific) << std::setprecision(2);
std::cout << "   Dofs:                    " << dof_handler_primal.n_dofs()<< std::endl;
std::cout << "   Exact error:             " << exact_error_local << std::endl;
double total_estimated_error = 0.0;
for (unsigned int k=0;k<error_indicators.size();k++)
  total_estimated_error += error_indicators(k);

// Take the absolute of the estimated error.
// However, we might check if the signs
// of the exact error and the estimated error are the same.
total_estimated_error = std::abs(total_estimated_error);

std::cout << "   Estimated error (prim): " << total_estimated_error << std::endl;

// From the JCAM paper: compute indicator indices to check
// effectivity of error estimator.
double total_estimated_error_absolute_values = 0.0;
for (unsigned int k=0;k<error_indicators.size();k++)
  total_estimated_error_absolute_values += std::abs(error_indicators(k));

// "ind" things were mainly for paper with Thomas Richter (Richter/Wick; JCAM, 2015)
//  std::cout << "   Estimated error (ind):  " << total_estimated_error_absolute_values << std::end

std::cout << "   Ieff:                    " << total_estimated_error/exact_error_local << std::endl
```

```
   //std::cout << "   Iind:                    " << total_estimated_error_absolute_values/exact_error_l



   // Write everything into a file
   //file.precision(3);
   file << std::setiosflags(std::ios::scientific) << std::setprecision(2);
   file << dof_handler_primal.n_dofs() << "\t";
   file << exact_error_local << "\t";
   file << total_estimated_error << "\t";
   file << total_estimated_error_absolute_values << "\t";
   file << total_estimated_error/exact_error_local << "\t";
   file << total_estimated_error_absolute_values/exact_error_local << "\n";
   file.flush();

   // Write everything into a file gnuplot
   file_gnuplot << std::setiosflags(std::ios::scientific) << std::setprecision(2);
   //file_gnuplot.precision(3);
   file_gnuplot << dof_handler_primal.n_dofs() << "\t";
   file_gnuplot << exact_error_local << "\t";
   file_gnuplot << total_estimated_error << "\t";
   file_gnuplot << total_estimated_error_absolute_values << "\n";
   file_gnuplot.flush();


   // end Block 3


   // Block 4
   return total_estimated_error;


   // end Block 4

}


// Refinement strategy and carrying out the actual refinement.
template<int dim>
double NSE_PU_DWR_Problem<dim>::refine_average_with_PU_DWR (const unsigned int refinement_cycle)
{
  // Step 1:
  // Obtain error indicators from PU DWR estimator
  double estimated_DWR_error = compute_error_indicators_a_la_PU_DWR (refinement_cycle);

  // Step 2: Choosing refinement strategy
  // Here: averaged refinement
  // Alternatives are in deal.II:
  // refine_and_coarsen_fixed_fraction for example
  for (Vector<float>::iterator i=error_indicators.begin();
       i != error_indicators.end(); ++i)
    *i = std::fabs (*i);
```

```
  const unsigned int  dofs_per_cell_pou   = fe_pou.dofs_per_cell;
  std::vector< unsigned int >  local_dof_indices(dofs_per_cell_pou);

  // Refining all cells that have values above the mean value
  double error_indicator_mean_value = error_indicators.mean_value();

  // Pou cell  later
  typename DoFHandler<dim>::active_cell_iterator
    cell = dof_handler_pou.begin_active(),
    endc = dof_handler_pou.end();

  double error_ind = 0.0;
  //1.1; for drag and lift and none mesh smoothing
  //5.0; for pressure difference and maximal mesh smoothing
  double alpha = 1.1;


  for (; cell!=endc; ++cell)
    {
      error_ind = 0.0;
      cell->get_dof_indices(local_dof_indices);

      for (unsigned int i=0; i<dofs_per_cell_pou; ++i)
{
  error_ind += error_indicators(local_dof_indices[i]);
}

      // For uniform (global) mesh refinement,
      // just comment the following line
      if (error_ind >  alpha * error_indicator_mean_value)
cell->set_refine_flag();
    }


  triangulation.execute_coarsening_and_refinement ();

  return estimated_DWR_error;

}
```

17.2.8 deal.II (C++) heat equation used in Section 12.3.12

In this section, we provide the programming code for Section 12.3.12. In general, we notice that it does not make sense to copy a code into some written document and print it. Sometimes, however this is useful, specifically for carefully explaining basic steps, which is the case here. For this reason, the code is on the one hand as short as possible, with only the necessary documentation, but also (hopefully) self-contained and self-explaining.

```
/*
 * Author: Thomas Wick (2014)
 * ICES, UT Austin, USA
 *
 * Features of this program:
 * -------------------------
 *    Computes the nonstationary heat equation in 2d and 3d
```

```
 *    A) Constant and/or nonconstant diffusion coefficient
 *    B) Constant and/or nonconstant right hand side f
 *    C) Homogeneous or nonhomogeneous Dirichlet conditions
 *    D) Projection of initial data
 *    E) Declaration of the 2d and 3d classes
 *    F) Possible extension: implement different timestepping schemes
 *       with a suitable choice of theta
 *
 * Leibniz University Hannover
 * November 04, 2021
 * Running under deal.II version dealii-9.1.1
 *
 * This program is originally based on the deal.II steps 4 and 5
 * in version 8.1.0
 * for implementing the Poisson problem
 * and we refer for Author and License information to those tutorial steps.
 * The time step loop implementation is a straightforward
 * extension; here it is based on the ideas presented in
 * T. Wick; ANS, Vol. 1, No. 1, pp. 1-19.
 *
 * License of this program: GNU Lesser General
 * Public License as published by the Free Software Foundation
 *
 *
 */

////////////////////////////////////////////////
// For explanations of the includes, we refer
// to the deal.II documentation
#include <deal.II/grid/tria.h>
#include <deal.II/dofs/dof_handler.h>
#include <deal.II/grid/grid_generator.h>
#include <deal.II/grid/tria_accessor.h>
#include <deal.II/grid/tria_iterator.h>
#include <deal.II/dofs/dof_accessor.h>
#include <deal.II/fe/fe_q.h>
#include <deal.II/dofs/dof_tools.h>
#include <deal.II/fe/fe_values.h>
#include <deal.II/base/quadrature_lib.h>
#include <deal.II/base/function.h>
#include <deal.II/numerics/vector_tools.h>
#include <deal.II/numerics/matrix_tools.h>
#include <deal.II/lac/vector.h>
#include <deal.II/lac/full_matrix.h>
#include <deal.II/lac/sparse_matrix.h>
#include <deal.II/lac/dynamic_sparsity_pattern.h>
#include <deal.II/lac/solver_cg.h>
#include <deal.II/lac/precondition.h>
#include <deal.II/grid/grid_refinement.h>
#include <deal.II/numerics/error_estimator.h>

#include <deal.II/numerics/data_out.h>
#include <fstream>
#include <iostream>
```

```
#include <deal.II/base/logstream.h>

using namespace dealii;

////////////////////////////////////////////////////
// Main class
template <int dim>
class Step_Heat
{
public:
  Step_Heat ();
  void run ();

private:
  void set_runtime_parameters ();
  void make_grid ();
  void setup_system();
  void assemble_system ();
  void solve ();
  void output_results (const unsigned int cycle) const;

  void compute_functionals();

  Triangulation<dim>   triangulation;
  FE_Q<dim>            fe;
  DoFHandler<dim>      dof_handler;

  SparsityPattern      sparsity_pattern;
  SparseMatrix<double> system_matrix;

  Vector<double>       solution, old_timestep_solution;
  Vector<double>       system_rhs;

  double timestep, theta, time;
  std::string time_stepping_scheme;
  unsigned int max_no_timesteps, timestep_number;

};

////////////////////////////////////////////////////
// 1. Initial values
// 2. Boundary values
// 3. Right hand side f
// 4. Function for coefficients


////////////////////////////////////////////////////
// Initial values

  template <int dim>
  class InitialValues : public Function<dim>
  {
    public:
      InitialValues () : Function<dim>(1) {}
```

```
      virtual double value (const Point<dim>   &p,
    const unsigned int  component = 0) const;

      virtual void vector_value (const Point<dim> &p,
 Vector<double>   &value) const;

  };


  template <int dim>
  double
  InitialValues<dim>::value (const Point<dim>  &p,
     const unsigned int component) const
  {
    // Choice 1
    //return 1.0;

    // Choice 2 TODO 2d !!!
    return (std::sin(p[0]) * std::sin(p[1]));
  }


  template <int dim>
  void
  InitialValues<dim>::vector_value (const Point<dim> &p,
    Vector<double>   &values) const
  {
    for (unsigned int comp=0; comp<this->n_components; ++comp)
      values (comp) = InitialValues<dim>::value (p, comp);
  }


////////////////////////////////////////////////////////
// Right hand side
// Info: not fully implemented, but rather directly in the assembly
template <int dim>
class BoundaryValues : public Function<dim>
{
public:
  BoundaryValues () : Function<dim>() {}

  virtual double value (const Point<dim>   &p,
                        const unsigned int  component = 0) const;
};



template <int dim>
double BoundaryValues<dim>::value (const Point<dim> &p,
                                   const unsigned int /*component*/) const
{
  return p.square();
}
```

```
//////////////////////////////////////////////////////////
// Right hand side
// Info: not fully implemented, but rather directly in the assembly
template <int dim>
class RightHandSide : public Function<dim>
{
public:
  RightHandSide () : Function<dim>() {}

  virtual double value (const Point<dim>   &p,
                        const unsigned int  component = 0) const;
};

template <int dim>
double RightHandSide<dim>::value (const Point<dim> &p,
                                  const unsigned int /*component*/) const
{
  double return_value = 0;
  for (unsigned int i=0; i<dim; ++i)
    return_value += 4*std::pow(p(i), 4);

  return return_value;
}


//////////////////////////////////////////////////////////
// Non-constant coefficient values for Laplacian
template <int dim>
class Coefficient : public Function<dim>
{
public:
  Coefficient ()  : Function<dim>() {}

  virtual double value (const Point<dim>   &p,
                        const unsigned int  component = 0) const;

  virtual void value_list (const std::vector<Point<dim> > &points,
                           std::vector<double>            &values,
                           const unsigned int              component = 0) const;
};



template <int dim>
double Coefficient<dim>::value (const Point<dim> &p,
                                const unsigned int /*component*/) const
{
  if (p[1] >= 0.0)
    return 20;
  else
```

```
   return 1;
}


template <int dim>
void Coefficient<dim>::value_list (const std::vector<Point<dim> > &points,
                                   std::vector<double>            &values,
                                   const unsigned int              component) const
{
  Assert (values.size() == points.size(),
          ExcDimensionMismatch (values.size(), points.size()));

  Assert (component == 0,
          ExcIndexRange (component, 0, 1));

  const unsigned int n_points = points.size();

  for (unsigned int i=0; i<n_points; ++i)
    {
      if (points[i][1] >= 0.0)
        values[i] = 20;
      else
        values[i] = 1;
    }
}



////////////////////////////////////////////////////////
// Constructor
template <int dim>
Step_Heat<dim>::Step_Heat ()
  :
  fe (1),
  dof_handler (triangulation)
{}


////////////////////////////////////////////////////////
// Definiting several model parameters
template <int dim>
void Step_Heat<dim>::set_runtime_parameters()
{
 // Timestepping schemes
  //BE, FE, CN
  time_stepping_scheme = "BE";

  // BE = 1.0,
  // FE = 0.0,
  // CN = 0.5,
  // F) Not yet implemented for the cases other than BE: 1.0
  theta = 0.5;

  // Timestep size:
```

```
  timestep = 0.125;

  // Maximum number of timesteps:
  max_no_timesteps = 200;

  // A variable to count the number of time steps
  timestep_number = 0;

  // Counts total time
  time = 0;


  // Goal functionals
  // BE:
  // L2 norm: 2.500000e+01   6.644876e-02
  // L2 norm: 2.500000e+01   1.236463e-01
  // L2 norm: 2.500000e+01   2.485528e-01

  // CN:
  // L2 norm: 2.500000e+01   2.352296e-05
  // L2 norm: 2.500000e+01   4.243702e-02
  // L2 norm: 2.500000e+01   1.660507e-01

  // Ref 6, CN
  //L2 norm: 2.500000e+01   5.975624e-06
  //L2 norm: 2.500000e+01   4.251649e-02
  //L2 norm: 2.500000e+01   1.662468e-01
  //L2 norm: 2.500000e+01   4.058831e-01

}


////////////////////////////////////////////////////////
// Initialize the mesh
template <int dim>
void Step_Heat<dim>::make_grid ()
{
  GridGenerator::hyper_cube (triangulation, 0, M_PI);

  // In the following, we read a *.inp grid from a file.
  /*
  std::string grid_name;
  grid_name  = "unit_square_1.inp";

  GridIn<dim> grid_in;
  grid_in.attach_triangulation (triangulation);
  std::ifstream input_file(grid_name.c_str());
  Assert (dim==2, ExcInternalError());
  grid_in.read_ucd (input_file);
  */

  triangulation.refine_global (6);

  std::cout << "   Number of active cells: "
```

```
            << triangulation.n_active_cells()
            << std::endl
            << "   Total number of cells: "
            << triangulation.n_cells()
            << std::endl;
}

////////////////////////////////////////////////////////
// Setting up sparsity pattern for system matrix, rhs, solution vector
template <int dim>
void Step_Heat<dim>::setup_system ()
{
  dof_handler.distribute_dofs (fe);

  std::cout << "   Number of degrees of freedom: "
            << dof_handler.n_dofs()
            << std::endl;

  //CompressedSparsityPattern c_sparsity(dof_handler.n_dofs());
  DynamicSparsityPattern c_sparsity(dof_handler.n_dofs());
  DoFTools::make_sparsity_pattern (dof_handler, c_sparsity);
  sparsity_pattern.copy_from(c_sparsity);

  system_matrix.reinit (sparsity_pattern);

  solution.reinit (dof_handler.n_dofs());
  old_timestep_solution.reinit (dof_handler.n_dofs());
  system_rhs.reinit (dof_handler.n_dofs());
}


////////////////////////////////////////////////////////
// This is the heart in which the
// discretized equations are locally (per element) assembled.
// Therein, local integrals are evaluated with Gauss quadrature.
template <int dim>
void Step_Heat<dim>::assemble_system ()
{
  QGauss<dim>  quadrature_formula(2);

  FEValues<dim> fe_values (fe, quadrature_formula,
                           update_values   | update_gradients |
                           update_quadrature_points | update_JxW_values);

  const unsigned int   dofs_per_cell = fe.dofs_per_cell;
  const unsigned int   n_q_points    = quadrature_formula.size();

  FullMatrix<double>   cell_matrix (dofs_per_cell, dofs_per_cell);
  Vector<double>       cell_rhs (dofs_per_cell);

  std::vector<types::global_dof_index> local_dof_indices (dofs_per_cell);

  // For previous time step solutions
  std::vector<double> old_timestep_solution_values (n_q_points);
```

```
  std::vector<std::vector<Tensor<1,dim> > >
    old_timestep_solution_grads (n_q_points,std::vector<Tensor<1,dim> > (1));

  // Right hand side
  const RightHandSide<dim> right_hand_side;

  // Class for nonconstant coefficients
  const Coefficient<dim> coefficient;
  std::vector<double>    coefficient_values (n_q_points);

  double density = 1.0;

  // Next, we again have to loop over all elements and assemble local
  // contributions.  Note, that an element is a quadrilateral in two space
  // dimensions, but a hexahedron in 3D.
  typename DoFHandler<dim>::active_cell_iterator
  cell = dof_handler.begin_active(),
  endc = dof_handler.end();

  for (; cell!=endc; ++cell)
    {
      fe_values.reinit (cell);
      cell_matrix = 0;
      cell_rhs = 0;

      // Old_timestep_solution values are interpolated on the present cell
      fe_values.get_function_values (old_timestep_solution, old_timestep_solution_values);
      fe_values.get_function_gradients (old_timestep_solution, old_timestep_solution_grads);


      // Update of nonconstant coefficients
      coefficient.value_list (fe_values.get_quadrature_points(),
                              coefficient_values);

      // Loop over all quadrature points (evaluating integrals numerically)
      for (unsigned int q_point=0; q_point<n_q_points; ++q_point)
{
  // Get FE-solution values for previous time step at present quadrature point
  const double old_timestep_u = old_timestep_solution_values[q_point];

  Tensor<1,dim> old_timestep_grad_u;
  old_timestep_grad_u[0] = old_timestep_solution_grads[q_point][0][0];
  old_timestep_grad_u[1] = old_timestep_solution_grads[q_point][0][1];


        for (unsigned int i=0; i<dofs_per_cell; ++i)
          {
            for (unsigned int j=0; j<dofs_per_cell; ++j)
      {
// Mass term from the time derivative
// Change indices such that (j,i) because test
// function determines the row in system matrix A
// See lecture notes in Chapter 6
```

```
cell_matrix(j,i) += density * (fe_values.shape_value (i, q_point) *
     fe_values.shape_value (j, q_point) *
     fe_values.JxW (q_point));


cell_matrix(j,i) += timestep * theta *
  (// A) Constant diffusion a=1:
   1.0 *
   // Nonconstant diffusion coefficient
   //coefficient_values[q_point] *
   fe_values.shape_grad (i, q_point) *
   fe_values.shape_grad (j, q_point) *
   fe_values.JxW (q_point));


      }

             cell_rhs(i) += (old_timestep_u * fe_values.shape_value (i, q_point)
    + fe_values.shape_value (i, q_point) *
                           // B) Constant right hand side f=0:
    0.0
    // Non constant right hand side:
    //right_hand_side.value (fe_values.quadrature_point (q_point))
    ) * fe_values.JxW (q_point);

    // Old timestep Laplace operator
    cell_rhs(i) -= timestep * (1.0 - theta) * 1.0 *
                   (old_timestep_grad_u * fe_values.shape_grad (i, q_point)
    ) * fe_values.JxW (q_point);



          }
}

      // Putting local element contributions into global system matrix A
      // and global right hand side vector b
      cell->get_dof_indices (local_dof_indices);
      for (unsigned int i=0; i<dofs_per_cell; ++i)
        {
          for (unsigned int j=0; j<dofs_per_cell; ++j)
            system_matrix.add (local_dof_indices[i],
                               local_dof_indices[j],
                               cell_matrix(i,j));

          system_rhs(local_dof_indices[i]) += cell_rhs(i);
        }
    }


  // Apply Dirichlet boundary conditions
  std::map<types::global_dof_index,double> boundary_values;
  VectorTools::interpolate_boundary_values (dof_handler,
                                            0,
```

```
    // C) Homogeneous Dirichlet conditions:
    ZeroFunction<dim>(),
    // Non-homogeneous Dirichlet conditions:
                                          //BoundaryValues<dim>(),
                                          boundary_values);
  MatrixTools::apply_boundary_values (boundary_values,
                                      system_matrix,
                                      solution,
                                      system_rhs);
}


//////////////////////////////////////////////////////
// Linear solution using the conjugate gradient
// method without preconditioning.
// See deal.II tutorials for extensions with preconditioners
// See also my lecture notes for extensions towards geometric multigrid
template <int dim>
void Step_Heat<dim>::solve ()
{
  SolverControl           solver_control (1000, 1e-12);
  SolverCG<>              solver (solver_control);
  solver.solve (system_matrix, solution, system_rhs,
                PreconditionIdentity());

  std::cout << "   " << solver_control.last_step()
            << " CG iterations needed to obtain convergence."
            << std::endl;
}

//////////////////////////////////////////////////////////
// Goal functional evaluations
// -> correctness of solution
// -> possible interest in terms of applications
template <int dim>
void Step_Heat<dim>::compute_functionals ()
{
  // Compute global L2 norm
  Vector<float> difference_per_cell(triangulation.n_active_cells());
  VectorTools::integrate_difference(dof_handler,
    solution,
    //Solution<dim>(),
    ZeroFunction<dim>(),
    difference_per_cell,
    QGauss<dim>(4),
    VectorTools::L2_norm);
  const double L2_error =
    VectorTools::compute_global_error(triangulation,
      difference_per_cell,
      VectorTools::L2_norm);


  // Compute point value
  Point<dim> p = Point<dim>(M_PI/2.0, M_PI/2.0);
```

```
  unsigned int component = 0;
  Vector<double> tmp_vector(1);
  VectorTools::point_value (dof_handler,
    solution,
    p,
    tmp_vector);

  // Output
  std::cout << "L2 norm:     " << time << "   " << std::scientific << L2_error << std::endl;
  std::cout << "Point value: " << time << "   " << p << "   " << tmp_vector(component) << std::endl;
  std::cout << std::endl;

}


////////////////////////////////////////////////////////
// Graphical output: here *.vtk
template <int dim>
void Step_Heat<dim>::output_results (const unsigned int cycle) const
{
  DataOut<dim> data_out;

  data_out.attach_dof_handler (dof_handler);
  data_out.add_data_vector (solution, "solution");

  data_out.build_patches ();

  std::string filename_basis = "solution_";
  std::ostringstream filename;
  std::cout << "-------------------" << std::endl;
  std::cout << "Write solution" << std::endl;
  std::cout << "-------------------" << std::endl;
  std::cout << std::endl;

  if (dim == 2)
    {
      filename << filename_basis
       << "2d_"
       << Utilities::int_to_string (cycle, 2)
       << ".vtk";
    }
  else
    {
      filename << filename_basis
       << "3d_"
       << Utilities::int_to_string (cycle, 2)
       << ".vtk";
    }

  std::ofstream output (filename.str().c_str());
  data_out.write_vtk (output);
}
```

```
//////////////////////////////////////////////////////
// Final method to run the program and designing the
// time step loop
template <int dim>
void Step_Heat<dim>::run ()
{
  std::cout << "Solving problem in " << dim << " space dimensions." << std::endl;

  set_runtime_parameters();
  make_grid();
  setup_system ();

  // D) Project (possibly nonhomogeneous) initial values
  {
      ConstraintMatrix constraints;
      constraints.close();

      VectorTools::project (dof_handler,
    constraints,
    QGauss<dim>(3),
    InitialValues<dim>(),
    solution
    );

      output_results (0);
    }


  // Timestep loop
  do
    {
      std::cout << "Timestep " << timestep_number
<< " (" << time_stepping_scheme
<< ")" <<    ": " << time
<< " (" << timestep << ")"
<< "\n=============================="
<< "=================================="
<< std::endl;

      std::cout << std::endl;

      // Solve for next time step
      old_timestep_solution = solution;
      assemble_system ();
      solve ();

      // Write solutions
      output_results (timestep_number+1);

      // Compute goal functionals
      compute_functionals();

      time += timestep;
      ++timestep_number;
```

```
    }
  while (timestep_number <= max_no_timesteps);



}


////////////////////////////////////////////////////////
// Final method to start the program
int main ()
{
  deallog.depth_console (0);

  // E) Computes first 2d and then 3d solution
  {
    Step_Heat<2> laplace_problem_2d;
    laplace_problem_2d.run ();
  }

//  {
//    Step_Heat<3> laplace_problem_3d;
//    laplace_problem_3d.run ();
//  }

  return 0;
}
```

17.2.9 deal.II (C++) Navier-Stokes illustrating nonlinear problems, Chapter 13

In this section, we show how a coupled nonlinear problem is assembled in a monolithic fashion in deal.II. The numerical solution is based on Newton's method, Section 13.12, and the Navier-Stokes FEM discretization Section 13.19. The following code snippets are taken from https://thomaswick.org/Codes/step_nse_Nov_9_2021.cc

```
// Looping over all elements
  for (; cell!=endc; ++cell)
    {
      fe_values.reinit (cell);
      local_matrix = 0;

      // Previous Newton iteration values
      fe_values.get_function_values (solution_primal, old_solution_values);
      fe_values.get_function_gradients (solution_primal, old_solution_grads);


  for (unsigned int q=0; q<n_q_points; ++q)
    {
      for (unsigned int k=0; k<dofs_per_cell; ++k)
{
  phi_i_v[k]       = fe_values[velocities].value (k, q);
  phi_i_grads_v[k] = fe_values[velocities].gradient (k, q);
  phi_i_p[k]       = fe_values[pressure].value (k, q);
}
```

```
      // We build values, vectors, and tensors
      // from information of the previous Newton step.
      // Required for convection term
      Tensor<1,dim> v;
      v[0] = old_solution_values[q](0);
      v[1] = old_solution_values[q](1);

      Tensor<2,dim> grad_v;
      grad_v[0][0] = old_solution_grads[q][0][0];
      grad_v[0][1] = old_solution_grads[q][0][1];
      grad_v[1][0] = old_solution_grads[q][1][0];
      grad_v[1][1] = old_solution_grads[q][1][1];


      // Outer loop for dofs
      for (unsigned int i=0; i<dofs_per_cell; ++i)
{
  Tensor<2, 2> pI_LinP;
  pI_LinP.clear();
  pI_LinP[0][0] = phi_i_p[i];
  pI_LinP[1][1] = phi_i_p[i];

  Tensor<2,dim> grad_v_LinV;
  grad_v_LinV[0][0] = phi_i_grads_v[i][0][0];
  grad_v_LinV[0][1] = phi_i_grads_v[i][0][1];
  grad_v_LinV[1][0] = phi_i_grads_v[i][1][0];
  grad_v_LinV[1][1] = phi_i_grads_v[i][1][1];


  const Tensor<2,dim>  stress_fluid_LinAll = -pI_LinP
    + density_fluid * viscosity * (phi_i_grads_v[i] + transpose(phi_i_grads_v[i]));

  const Tensor<1,dim> convection_fluid_LinAll = density_fluid * (phi_i_grads_v[i] * v + grad_v * phi_

  const double incompressibility_LinAll = phi_i_grads_v[i][0][0] + phi_i_grads_v[i][1][1];

  // Inner loop for dofs
  for (unsigned int j=0; j<dofs_per_cell; ++j)
    {
      // Fluid , NSE
      const unsigned int comp_j = fe_primal.system_to_component_index(j).first;
      if (comp_j == 0 || comp_j == 1)
{
  local_matrix(j,i) += (convection_fluid_LinAll * phi_i_v[j] +
scalar_product(stress_fluid_LinAll, phi_i_grads_v[j])
) * fe_values.JxW(q);
}
      else if (comp_j == 2)
{
  local_matrix(j,i) += (incompressibility_LinAll *  phi_i_p[j]
) * fe_values.JxW(q);
}
      // end j dofs
```

```
    }
  // end i dofs
}
      // end n_q_points
    }

          // This is the same as discussed in step-22:
  cell->get_dof_indices (local_dof_indices);
  constraints_primal.distribute_local_to_global (local_matrix, local_dof_indices,
  system_matrix_primal);


    }   // end cell


//////////////////////////////////////////////////////
...
//////////////////////////////////////////////////////

  for (; cell!=endc; ++cell)
    {
      fe_values.reinit (cell);
      local_rhs = 0;

      // Previous Newton iteration
      fe_values.get_function_values (solution_primal, old_solution_values);
      fe_values.get_function_gradients (solution_primal, old_solution_grads);


  for (unsigned int q=0; q<n_q_points; ++q)
    {
      Tensor<2, 2> pI;
      pI.clear();
      pI[0][0] = old_solution_values[q](dim);
      pI[1][1] = old_solution_values[q](dim);

      Tensor<1,dim> v;
      v[0] = old_solution_values[q](0);
      v[1] = old_solution_values[q](1);


      Tensor<2,dim> grad_v;
      grad_v[0][0] = old_solution_grads[q][0][0];
      grad_v[0][1] = old_solution_grads[q][0][1];
      grad_v[1][0] = old_solution_grads[q][1][0];
      grad_v[1][1] = old_solution_grads[q][1][1];

      Tensor<2,dim> sigma_fluid;
      sigma_fluid.clear();
      sigma_fluid = -pI + density_fluid * viscosity * (grad_v + transpose(grad_v));


      // Divergence of the fluid
      const double incompressiblity_fluid = grad_v[0][0] + grad_v[1][1];
```

```
      // Convection term of the fluid
      Tensor<1,dim> convection_fluid;
      convection_fluid.clear();
      convection_fluid = density_fluid * grad_v * v;

      for (unsigned int i=0; i<dofs_per_cell; ++i)
{
  // Fluid, NSE
  const unsigned int comp_i = fe_primal.system_to_component_index(i).first;
  if (comp_i == 0 || comp_i == 1)
    {
      const Tensor<1,dim> phi_i_v = fe_values[velocities].value (i, q);
      const Tensor<2,dim> phi_i_grads_v = fe_values[velocities].gradient (i, q);

      local_rhs(i) -= (convection_fluid * phi_i_v +
       scalar_product(sigma_fluid, phi_i_grads_v)
       ) *  fe_values.JxW(q);

    }
  else if (comp_i == 2)
    {
      const double phi_i_p = fe_values[pressure].value (i, q);
      local_rhs(i) -= (incompressiblity_fluid * phi_i_p) *  fe_values.JxW(q);
    }
  // end i dofs
}
      // close n_q_points
    }

          cell->get_dof_indices (local_dof_indices);
  constraints_primal.distribute_local_to_global (local_rhs, local_dof_indices,
  system_rhs_primal);


    }  // end cell
```

17.3 Chapter summary and outlook

We finish this booklet in the next chapter by posing some central questions that help to recapitulate the important topics of this lecture.

18 Wrap-up

18.1 Quiz

18.1.1 Basic techniques

1. Define the Laplace operator.
2. Write down the formula of partial integration in higher dimensions!
3. State the Poisson problem with homogeneous Dirichlet conditions. Write this problem in a variational form using the correct function spaces.
4. Define the H^1 space.
5. Define the H_0^1 space.
6. What are difficulties of H^1 functions in higher dimensions $n \geq 2$?
7. What is a Hilbert space?
8. What are possible boundary conditions for second-order boundary value problems?
9. What is the idea of the FEM?
10. Why do we need numerical quadrature in finite element calculations?
11. What are the three characterizing properties of an finite element?
12. Coming from the variational form, give the discrete form and construct a finite element scheme of the Poisson problem until the final final linear equation system.
13. Explain the idea of Finite Differences (FD).
14. Discretize the 1D Poisson problem with FD.
15. What is the difference of FD and the FEM?
16. What is the discretization error?
17. What is the maximum principle?
18. What is an M matrix?
19. For what is the M matrix property important?
20. How can we characterize elliptic, parabolic, and hyperbolic problems?
21. What is the difference between a priori and a posteriori error estimates?
22. Let $h \to 0$. What does this mean for $\|u - u_h\|$ (FEM) and for the solution of $Ax = b$, where A is the usual stiffness matrix of Poisson's problem?
23. Formulate Poisson's problem with Dirichlet, Neumann, Robin conditions, respectively.
24. How is well-posedness of problems defined?
25. What is the Poisson problem? How do we obtain existence, uniqueness, and continuous dependence on the right hand side (Hint: weak form)?
26. What is the Lax-Milgram lemma? What are the assumptions?
27. Outline the key ideas of the proof of the Lax-Milgram lemma.

28. Give a physical interpretation of the Poisson problem.
29. Why are all integrals evaluated on a master element in FEM?
30. Write down the loops to assemble a finite element system matrix.
31. Why is it sufficient to use the Riesz lemma to prove existence of the Poisson problem rather than using the Lax-Milgram lemma?
32. What are challenges investigating PDEs in higher dimensions $d \geq 2$?
33. What is a best approximation?
34. What is Galerkin orthogonality? And what is its geometrical interpretation?
35. What are interpolation error estimates?
36. What are the ingredients (type of error estimates) to obtain convergence results in FEM?
37. How do we get an improved estimate for the L^2 norm for the Poisson problem using finite elements?
38. Which result is better: $\|u - u_h\|_{L^2} = O(h^2)$ or $\|u - u_h\|_{H^1} = O(h)$?
39. What is the Aubin-Nitsche trick?
40. What does the Céa lemma say?
41. How can we solve the linear equation systems $Ax = b$?
42. What is a preconditioner?
43. Give a stopping criterion for an iterative scheme!
44. Why is the condition number important?
45. What is the condition number of the system matrix of the Poisson problem?
46. Describe the basic idea of multigrid
47. What is a V cycle?
48. What is a smoother?
49. What is the optimal arithmetic complexity of a geometric multigrid method?
50. Formulate the heat / wave equation and discretize in space and time.
51. Give examples of different time-stepping schemes. What schemes exist and what are their principles differences?
52. Explain the key ideas to obtain stability estimates for parabolic problems.
53. What is the principle idea to proof energy conservation of the wave equation on the time-discrete level?
54. Why are implicit schemes mostly preferred in time-dependent problems?
55. What can cause instabilities in temperal problems and how can they be cured?
56. Why are Crank-Nicolson-type schemes (nearly) optimal schemes for wave-like problems?
57. Are explicit time-stepping schemes conditionally or unconditionally stable?
58. What is the Rothe method?
59. What is the method of lines?
60. What is the idea of a full space-time Galerkin discretization?

18.1.2 Advanced techniques

1. What is useful when employing goal-oriented error estimation? What is a goal functional?
2. What is the adjoint?
3. Why is the adjoint our friend?
4. Formulate Newton's method in $\mathbb{R}$.
5. Why do we need a posteriori error estimation?
6. Why do we need mesh adaptivity?
7. Define the directional derivative!
8. Formulate Newton's method for variational problems
9. What are examples of vector-valued problems? Id est: what is a vector-valued problem?
10. How are linear and quadratic convergence characterized in practice?
11. Give an example of a nonlinear equation.
12. What are numerical possiblities to solve nonlinear equations?
13. Explain a fixed-point iteration for solving nonlinear problems.
14. What is a coupled problem?
15. Which coupling versions exist?
16. How can we decouple a coupled problem? What are the advantages and shortcomings?
17. Why are interface-couplings more difficult than volume coupling?
18. What are numerical methods for interface coupling?
19. Given PDE1 and PDE2, formulate a compact semi-linear form for a monolithic version.
20. Formulate Newton's method for a variational-monolithic coupled problem
21. What are the advantages or shortcomings of staggered approaches?
22. What is the crucial point in terms of the numerical solution, when several PDEs are coupled in a monolithic fashion?

18.2 Hints for presenting results in a talk (FEM-C++ practical class)

We collect some hints to make good presentations using, for instance, beamer latex or PowerPoint:

1. Structure your talk clearly: what is the goal? How did you manage to reach this goal? What are your results? What is your interpretation? Why did you choose specific methods (advantages / shortcomings in comparison to other existing techniques)?
2. Title page: title, author names, co-authors (if applicable), your institute/company, name of event you are talking
3. Page 2: list of contents
4. Conclusions at the end: a few take-home messages
5. Rule of thumb: one minute per slide. For a 20 minutes presentation a good mean value is to have 20 slides.
6. A big risk of beamer/Power-Point presentations is that we rush over the slides. You are the expert and know everything! But your audience is not. Give the audience time to read the slide!
7. Taylor your talk to the audience!
8. Put a title on each slide!
9. In plots and figures: make lines, the legend, and axes big enough such that people from the back can still read your plots. A good **no go, bad** example is my Figure 14 in which the legend is by far too small. Do not do that! A good example is Figure 31.
10. Tables with numbers: if you create columns of numbers do not put all of them. If you have many, you may mark the most important in a different color to highlight them.
11. Do not glue to your slides. Despite you have everything there, try to explain things in a free speech.
12. Less is more:
    a) Do not put too much text on each slide!
    b) Do not write full sentences, but use bullet points!
    c) Do not use too fancy graphics charts, plots, etc, where everything is moving, etc.
    d) Do not try to pack everything you have learned into one presentation. Everybody believes that you know many, many things. The idea of a presentation is to present in a clear way a piece of your work!
13. As always: the first sentence is the most difficult one: practice for yourself some nice welcome words to get smoothly started.
14. Practice your final presentation ahead of time either alone, with friends or colleagues.
15. Just be authentic during the presentation.
16. If you easily become nervous, avoid coffee before the presentation.

18.2.1 Writing an abstract

Before presenting scientific results at workshops or conferences, usually an abstract is required, which summarizes in a consise way your results. An example from myself is provided in this section. [15]

Fluid-Structure Interaction in Arbitrary Lagrangian-Eulerian and Fully Eulerian Coordinates

A clear title

Advances in Computational Mechanics Conference, San Diego, USA
Feb 27, 2013

Name of event
Date

Thomas Wick,
Institute for Computational Engineering and Sciences,
The University of Texas at Austin,
email: twick@ices.utexas.edu

Highlight presenting author

Author 1: Name, application, email

Thomas Richter,
Institute for Applied Mathematics,
Heidelberg University,
email: thomas.richter@iwr.uni-heidelberg.de

Author 2: "

Abstract

In this talk, we address two formulations for fluid-structure interaction. Traditionally, the well-established ALE (Arbitrary Lagrangian-Eulerian) approach provides a simple procedure to couple fluid equations with solid deformations. In such a setting, the fluid equations are re-written in a new coordinate system such that they match at the interface with the solid. However, for large structural deformations or contact, the mesh moving becomes the critical part with a degeneration of the governing mapping. To overcome the deficiency, we present the novel fully Eulerian approach. The idea is the opposite way to the ALE method. The fluid equations are kept in their natural coordinates and the solid is transformed into the Eulerian framework. However, the major challenges are interface intersections in mesh elements and low-regularity hyperbolic transport terms. We propose methodologies how to cope with these deficiencies. Time discretization is based on stabilized second-order finite difference schemes whereas the spatial discretization is done with adaptive Galerkin finite elements. For the Navier-Stokes part, we employ inf-sup stable elements. The arising nonlinear problems are treated with Newton's method in which backtracking line search is adopted for the globalization of the convergence radius. The inner linear problems are solved with the Unsymmetric MultiFrontal method (UMFPACK), which is a direct method. The performance of our techniques is demonstrated with the help of numerical examples in which computational convergence is shown by computing goal functionals on three mesh levels and three different time step sizes.

What?

Why a new method?

What?

How: including analysis, verification and interpretation

Why $\hat{=}$ Purpose of this talk!
What $\hat{=}$ what is done?
How $\hat{=}$ How is the hypothesis proofed?
How is the problem treated? Which methodology?
How are the results justified and what is the interpretation?

1

[15] Thanks to Sebastian Kinnewig who had this idea to ask for such abstracts in our courses on FEM-C++ project work.

18.2.2 Some example pages

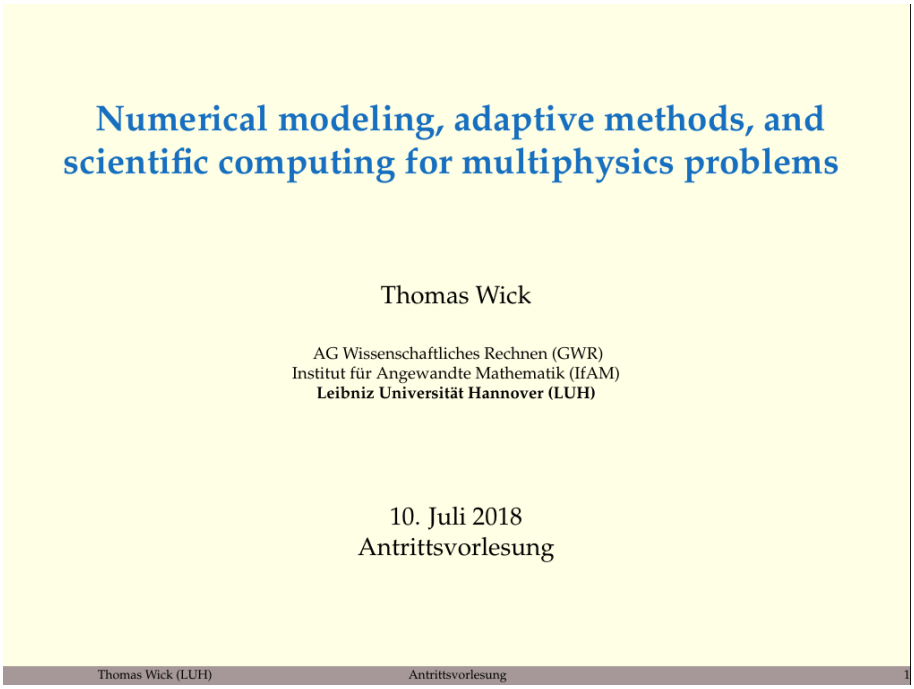


Figure 74: Title page. In the footline, the name, topic, and page number is displayed. Page numbers help the audience to point more precisely to a specific page when the talk is open for questions.

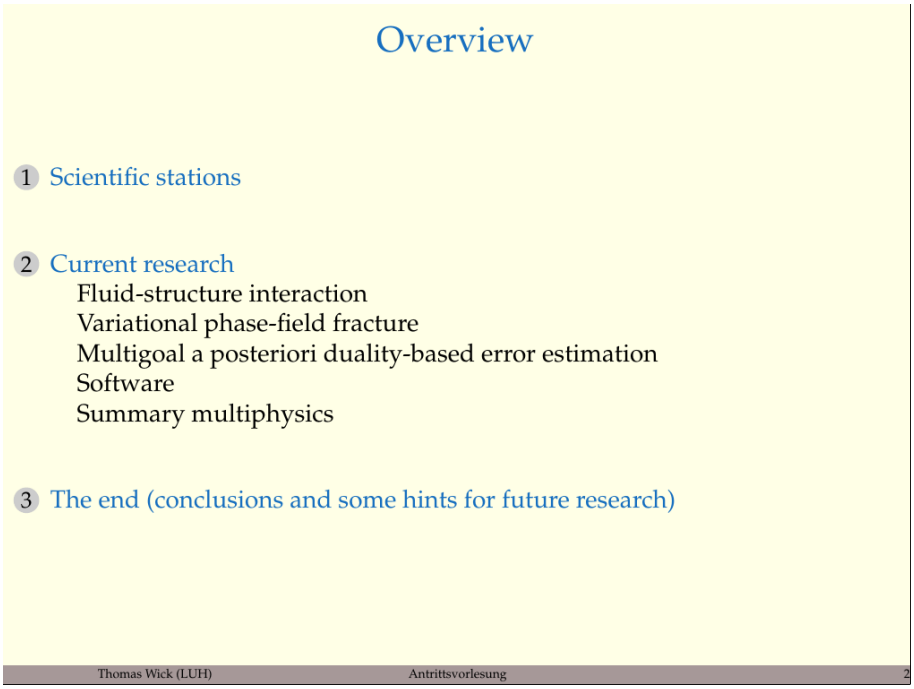


Figure 75: Page 2.

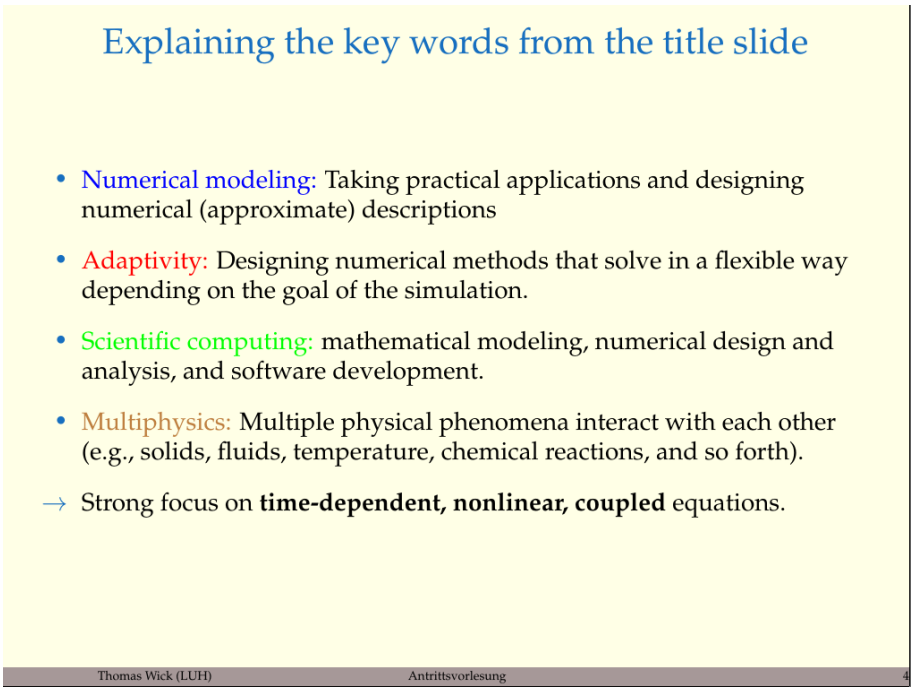


Figure 76: Some page with precise statements; not overpacked, bullet points, if not too exhausting colors may be used, ...

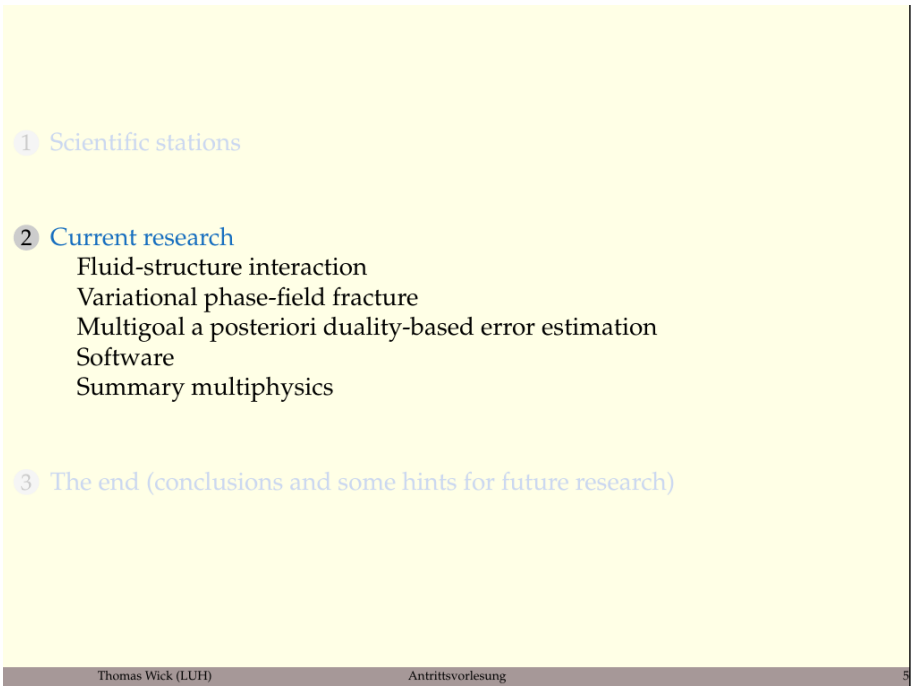
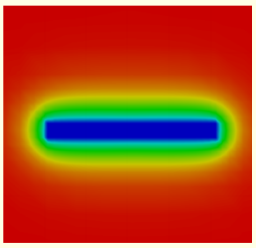


Figure 77: When switching to a new section, you may want to display the table of contents again to focus on this new section.

High performance parallel computing with MPI and local mesh adaptivity



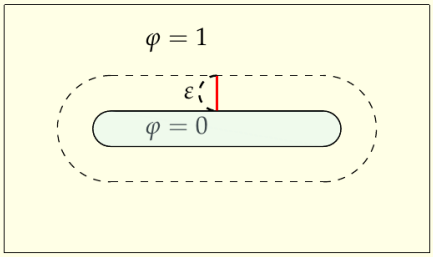


Figure : Sneddon's 3D stationary pressurized fracture: Fracture representation using the phase field φ. The inner blue region indicates the crack with $\varphi = 0$ and the red region indicates the unbroken zone where $\varphi = 1$. We have a diffusive zone in between (green region).

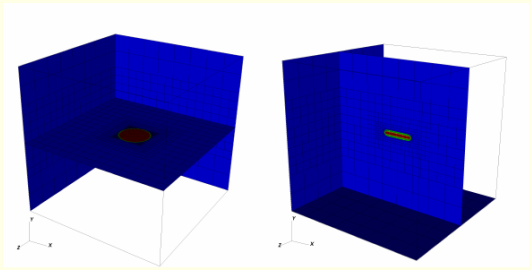
- Analytical derivations in Sneddon/Lowengrub 1969, Section 2.4 (2D) and Section 3.3 (3D); fracture does not propagate but only varies in its width;
- Goal: Compute a **goal functional value**. Let's say the total crack volume (TCV).

Thomas Wick (LUH) Antrittsvorlesung 37

Figure 78: Some page representing the configuration, goals, and references of a numerical example.

3D results for TCV[2]

ref	$10 l_0/h$	# DoFs	eps	TCV	Error
1	2	500	1.7321E+01	7.4519E-02	1355.44%
⋮					
8	256	946,852	1.3532E-01	5.8082E-03	13.44%
9	512	2,938,012	6.7658E-02	5.3764E-03	5.01%
10	1024	9,318,916	3.3829E-02	5.2131E-03	1.82%
11	2048	30,330,756	1.6915E-02	5.1567E-03	0.72%
12	4096	100,459,828	8.4573E-03	5.1352E-03	0.30%
Sneddon (exact)				5.1200E-03	



[2]Heister/Wick; 2018; also recent computations up to 50M DoFs by Katrin Mang

Thomas Wick (LUH) Antrittsvorlesung 38

Figure 79: Some page with results. If too many lines in a table, please use ... (dots). A reference to published literature can be added as well.

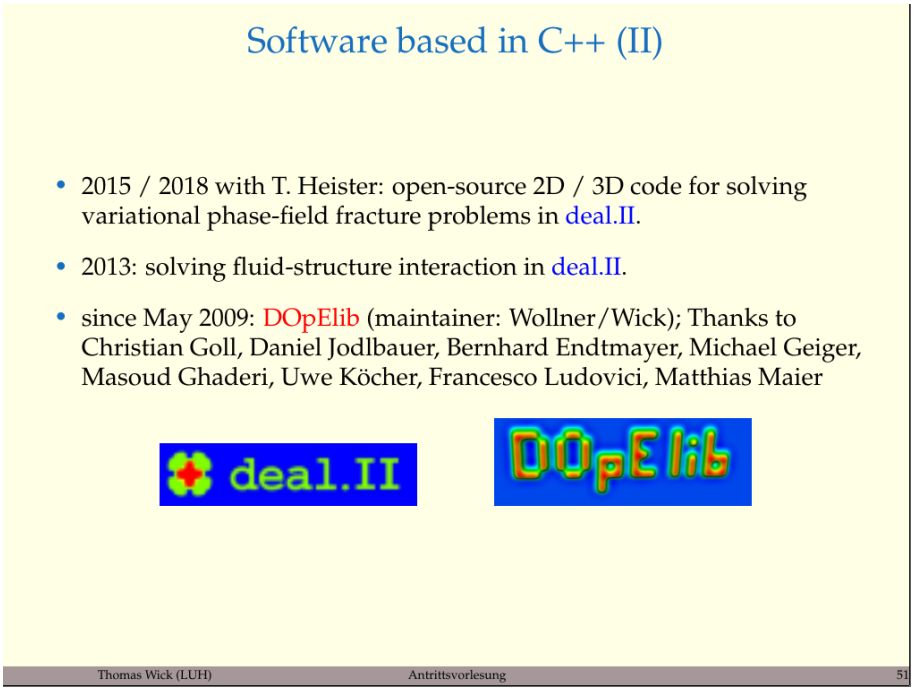


Figure 80: Some other page.

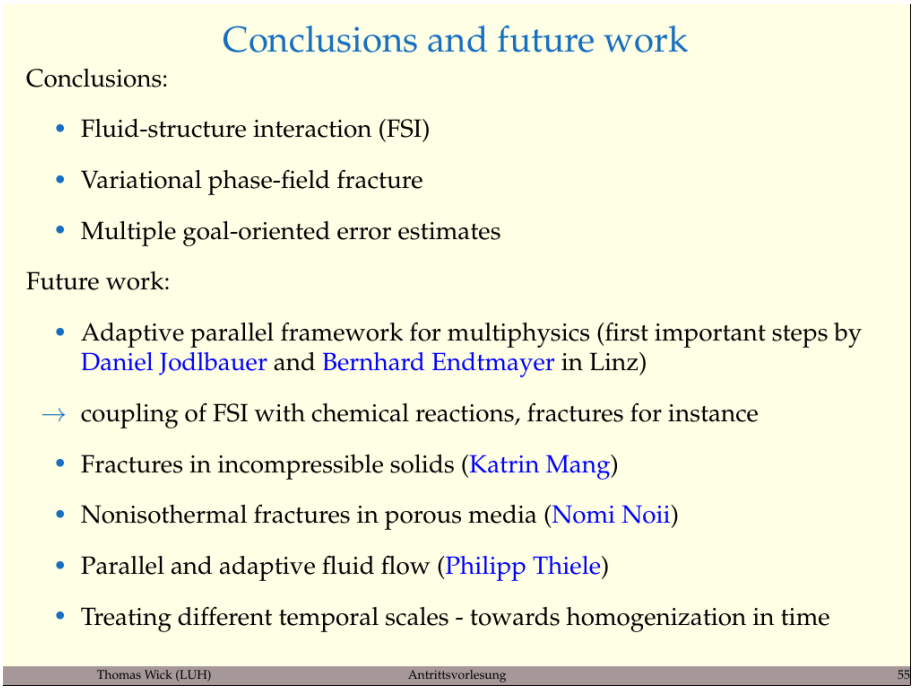


Figure 81: Conclusions with short and precise take-home messages. If applicable, you may want to add ongoing or future work.

18.3 The end

Eduard Mörike (1804 - 1875),
deutscher Erzähler, Lyriker und Dichter:
Septembermorgen

Im Nebel ruhet noch die Welt,
noch träumen Wald und Wiesen;
bald siehst du, wenn der Schleier fällt,
den blauen Himmel unverstellt,
herbstkräftig die gedämpfte Welt
in warmem Golde fließen.

Gottfried Wilhelm Leibniz (1646 - 1716):
“Es ist unwürdig, die Zeit von hervorragenden Leuten mit knechtischen Rechenarbeiten zu verschwenden, weil bei Einsatz einer Maschine auch der Einfältigste die Ergebnisse sicher hinschreiben kann.”
Gottfried Wilhelm Leibniz erfand das Staffelwalzenprinzip, welches eine große Errungenschaft beim Bau von mechanischen Rechenmaschinen darstellte, und gewissermaßen wegbereitend für unsere heutigen Rechnenmaschinen war: die Computer, welche einen Schwerpunkt in der modernen numerischen Mathematik und des Wissenschaftlichen Rechnens einnehmen.

References

[1] R. A. Adams. *Sobolev Spaces*. Academic Press, 1975.

[2] G. Allaire. *An Introduction to Mathematical Modelling and Numerical Simulation*. Oxford University Press, 2007.

[3] G. Allaire and F. Alouges. Analyse variationnelle des équations aux dérivées partielles. MAP 431: Lecture notes at Ecole Polytechnique, 2016.

[4] H. W. Alt. *Lineare Funktionalanalysis*. Springer, Berlin-Heidelberg, 5nd edition, 2006.

[5] S. Amstutz and T. Wick. Refresher course in maths and a project on numerical modeling done in twos. Hannover : Institutionelles Repositorium der Leibniz Universität Hannover, DOI: https://doi.org/10.15488/11629, December 2021.

[6] D. Arndt, W. Bangerth, D. Davydov, T. Heister, L. Heltai, M. Kronbichler, M. Maier, J.-P. Pelteret, B. Turcksin, and D. Wells. The `deal.II` library, version 8.5. *Journal of Numerical Mathematics*, 2017.

[7] W. Arnoldi. The principle of minimized iteration in the solution of the matrix eigenvalue problem. *Quart. Appl. Math.*, 9:17–29, 1951.

[8] W. Bangerth, M. Geiger, and R. Rannacher. Adaptive Galerkin finite element methods for the wave equation. *Comput. Methods Appl. Math.*, 10:3–48, 2010.

[9] W. Bangerth, R. Hartmann, and G. Kanschat. deal.II – a general purpose object oriented finite element library. *ACM Trans. Math. Softw.*, 33(4):24/1–24/27, 2007.

[10] W. Bangerth, T. Heister, and G. Kanschat. *Differential Equations Analysis Library*, 2012.

[11] W. Bangerth and R. Rannacher. *Adaptive Finite Element Methods for Differential Equations*. Birkhäuser, Lectures in Mathematics, ETH Zürich, 2003.

[12] S. Bartels. *Numerical Methods for Nonlinear Partial Differential Equations*. Springer, 2015.

[13] P. Bastian. Lecture notes on scientific computing with partial differential equations. Vorlesungsskriptum, 2014.

[14] R. Becker and R. Rannacher. A feed-back approach to error control in finite element methods: basic analysis and examples. *East-West J. Numer. Math.*, 4:237–264, 1996.

[15] R. Becker and R. Rannacher. An optimal control approach to a posteriori error estimation in finite element methods. *Acta Numerica, Cambridge University Press*, pages 1–102, 2001.

[16] A. Bensoussan, J. Lions, and G. Papanicolaou. *Asymptotic Analysis for Periodic Structures*. North-Holland, 1978.

[17] C. Bernardi and E. Süli. Time and space adaptivity for the second-order wave equation. *Math. Models Methods Appl. Sci.*, 15(2):199–225, 2005.

[18] M. Besier. *Adaptive Finite Element methods for computing nonstationary incompressible Flows*. PhD thesis, University of Heidelberg, 2009.

[19] M. Besier and R. Rannacher. Goal-oriented space-time adaptivity in the finite element galerkin method for the computation of nonstationary incompressible flow. *Int. J. Num. Meth. Fluids*, 70:1139–1166, 2012.

[20] M. Biot. Consolidation settlement under a rectangular load distribution. *J. Appl. Phys.*, 12(5):426–430, 1941.

[21] M. Biot. General theory of three-dimensional consolidation. *J. Appl. Phys.*, 12(2):155–164, 1941.

[22] M. Biot. Theory of elasticity and consolidation for a porous anisotropic solid. *J. Appl. Phys.*, 25:182–185, 1955.

[23] M. Braack and A. Ern. A posteriori control of modeling errors and discretization errors. *Multiscale Model. Simul.*, 1(2):221–238, 2003.

[24] D. Braess. *Finite Elemente.* Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg, vierte, überarbeitete und erweiterte edition, 2007.

[25] M. Braun. *Differential equations and their applications.* Springer, 1993.

[26] S. C. Brenner and L. R. Scott. *The mathematical theory of finite element methods.* Number 15 in Texts in applied mathematics ; 15 ; Texts in applied mathematics. Springer, New York, NY, 3. ed. edition, 2008.

[27] H. Brezis. *Analyse Fonctionnelle, Theorie et Applications.* Masson Paris, 1983.

[28] H. Brezis. *Functional analysis, Sobolev spaces and partial differential equations.* Springer New York, Dordrecht, Heidelberg, London, 2011.

[29] M. Bristeau, R. Glowinski, and J. Periaux. Numerical methods for the Navier-Stokes equations. *Comput. Phys. Rep.*, 6:73–187, 1987.

[30] G. F. Carey and J. T. Oden. *Finite Elements. Volume III. Compuational Aspects.* The Texas Finite Element Series, Prentice-Hall, Inc., Englewood Cliffs, 1984.

[31] C. Carstensen, M. Feischl, M. Page, and D. Praetorius. Axioms of adaptivity. *Computers and Mathematics with Applications*, 67(6):1195 – 1253, 2014.

[32] C. Carstensen and R. Verfürth. Edge residuals dominate a posteriori error estimates for low order finite element methods. *SIAM J. Numer. Anal.*, 36(5):1571–1587, 1999.

[33] S.-S. Chow. Finite element error estimates for non-linear elliptic equations of monotone type. *Numer. Math.*, 54:373–393, 1989.

[34] P. G. Ciarlet. *Mathematical Elasticity. Volume 1: Three Dimensional Elasticity.* North-Holland, 1984.

[35] P. G. Ciarlet. *The finite element method for elliptic problems.* North-Holland, Amsterdam [u.a.], 2. pr. edition, 1987.

[36] P. G. Ciarlet. *Linear and Nonlinear Functional Analysis with Applications.* SIAM, 2013.

[37] R. Courant. Variational methods for the solution of problems of equilibrium and vibrations. *Bull. Amer. Math. Soc.*, 49:1–23, 1943.

[38] H. Darcy. *Les Fontaines de la Ville de Dijon.* Dalmont, Paris, 1856.

[39] R. Dautray and J.-L. Lions. *Mathematical Analysis and Numerical Methods for Science and Technology*, volume 5. Springer-Verlag, Berlin-Heidelberg, 2000.

[40] P. Deuflhard. *Newton Methods for Nonlinear Problems*, volume 35 of *Springer Series in Computational Mathematics.* Springer Berlin Heidelberg, 2011.

[41] The Differential Equation and Optimization Environment: DOpElib. `http://www.dopelib.net`.

[42] W. Dörfler. A convergent adaptive algorithm for poisson's equation. *SIAM J. Numer. Anal.*, 33(3):1106–1124, 1996.

[43] G. Duvaut and J. L. Lions. *Inequalities in Mechanics and Physics.* Springer, Berlin-Heidelberg-New York., 1976.

[44] J. W. Eaton, D. Bateman, S. Hauberg, and R. Wehbring. *GNU Octave version 3.8.1 manual: a high-level interactive language for numerical computations*. CreateSpace Independent Publishing Platform, 2014. ISBN 1441413006.

[45] B. Endtmayer, U. Langer, and T. Wick. Multigoal-Oriented Error Estimates for Non-linear Problems. *Journal of Numerical Mathematics*, 27(4):215–236, 2019.

[46] B. Endtmayer, U. Langer, and T. Wick. Two-Side a Posteriori Error Estimates for the Dual-Weighted Residual Method. *SIAM J. Sci. Comput.*, 42(1):A371–A394, 2020.

[47] B. Endtmayer and T. Wick. A partition-of-unity dual-weighted residual approach for multi-objective goal functional error estimation applied to elliptic problems. *Computational Methods in Applied Mathematics*, 17(2):575–599, 2017.

[48] K. Eriksson, D. Estep, P. Hansbo, and C. Johnson. *Computational Differential Equations*. Cambridge University Press, 2009. http://www.csc.kth.se/ jjan/private/cde.pdf.

[49] D. Etling. *Theoretische Meteorologie*. Springer, 2008.

[50] L. C. Evans. *Partial differential equations*. American Mathematical Society, 2000.

[51] L. C. Evans. *Partial differential equations*. American Mathematical Society, 2010.

[52] T. Fliessbach. *Mechanik*. Spektrum Akademischer Verlag, 2007.

[53] S. Frei. *Eulerian finite element methods for interface problems and fluid-structure interactions*. PhD thesis, Heidelberg University, 2016.

[54] S. Frei and T. Richter. A locally modified parametric finite element method for interface problems. *SIAM Journal on Numerical Analysis*, 52(5):2315–2334, 2014.

[55] X. Gai. *A coupled geomechanics and reservoir flow model on parallel computers*. PhD thesis, The University of Texas at Austin, 2004.

[56] V. Girault, G. Pencheva, M. F. Wheeler, and T. Wildey. Domain decomposition for poroelasticity and elasticity with dg jumps and mortars. *Mathematical Models and Methods in Applied Sciences*, 21(1):169–213, 2011.

[57] V. Girault and P.-A. Raviart. *Finite Element method for the Navier-Stokes equations*. Number 5 in Computer Series in Computational Mathematics. Springer-Verlag, 1986.

[58] R. Glowinski and J. Periaux. Numerical methods for nonlinear problems in fluid dynamics. In *Proc. Intern. Seminar on Scientific Supercomputers*. North Holland, Feb. 2-6 1987.

[59] R. Glowinski and P. L. Tallec. *Augmented Lagrangian and Operator-Splitting Methods in Nonlinear Mechanics*. SIAM Stud. Appl. Math. 9. SIAM, Philadelphia, 1989.

[60] J. Goldenstein and J. Grashorn. FEM-Implementierung der instationären Wärmeleitungsgleichung in C++. Project in the FEM-C++ class, Institut für Angewandte Mathematik, Leibniz Universität Hannover, July 2018.

[61] H. Goldstein, C. P. P. Jr., and J. L. S. Sr. *Klassische Mechanik*. Wiley-VCH, 3. auflage edition, 2006.

[62] C. Goll, T. Wick, and W. Wollner. DOpElib: Differential equations and optimization environment; A goal oriented software library for solving pdes and optimization problems with pdes. *Archive of Numerical Software*, 5(2):1–14, 2017.

[63] P. Grisvard. *Elliptic Problems in Nonsmooth Domains*, volume 24. Pitman Advanced Publishing Program, Boston, 1985.

[64] C. Großmann and H.-G. Roos. *Numerische Behandlung partieller Differentialgleichungen*. Teubner-Studienbücher Mathematik ; Lehrbuch Mathematik. Teubner, Wiesbaden, 3., völlig überarb. und erw. aufl. edition, 2005.

[65] C. Großmann, H.-G. Roos, and M. Stynes. *Numerical Treatment of Partial Differential Equations*. Springer, 2007.

[66] H. Guzman. *Domain Decomposition Methods in Geomechanics*. PhD thesis, The University of Texas at Austin, 2012.

[67] W. Hackbusch. *Multi-Grid Methods and Applications*. Springer, 1985.

[68] W. Hackbusch. *Theorie und Numerik elliptischer Differentialgleichungen*. Vieweg+Teubner Verlag, 1986.

[69] G. H"ammerlin and K.-H. Hoffmann. *Numerische Mathematik*. Springer Verlag, 1992.

[70] M. Hanke-Bourgeois. *Grundlagen der numerischen Mathematik und des Wissenschaftlichen Rechnens*. Vieweg-Teubner Verlag, 2009.

[71] R. Hartmann. Multitarget error estimation and adaptivity in aerodynamic flow simulations. *SIAM Journal on Scientific Computing*, 31(1):708–731, 2008.

[72] R. Hartmann and P. Houston. Goal-oriented a posteriori error estimation for multiple target functionals. In T. Hou and E. Tadmor, editors, *Hyperbolic Problems: Theory, Numerics, Applications*, pages 579–588. Springer Berlin Heidelberg, 2003.

[73] T. Heister, M. F. Wheeler, and T. Wick. A primal-dual active set method and predictor-corrector mesh adaptivity for computing fracture propagation using a phase-field approach. *Comp. Meth. Appl. Mech. Engrg.*, 290:466 – 495, 2015.

[74] J. G. Heywood and R. Rannacher. Finite-element approximation of the nonstationary Navier-Stokes problem part iv: Error analysis for second-order time discretization. *SIAM Journal on Numerical Analysis*, 27(2):353–384, 1990.

[75] J. G. Heywood, R. Rannacher, and S. Turek. Artificial boundaries and flux and pressure conditions for the incompressible Navier-Stokes equations. *International Journal of Numerical Methods in Fluids*, 22:325–352, 1996.

[76] G. Holzapfel. *Nonlinear Solid Mechanics: A continuum approach for engineering*. John Wiley and Sons, LTD, 2000.

[77] J. Hron and S. Turek. *Proposal for numerical benchmarking of fluid-structure interaction between an elastic object and laminar incompressible flow*, volume 53, pages 146 – 170. Springer-Verlag, 2006.

[78] T. Hughes. *The finite element method*. Dover Publications, 2000.

[79] T. Hughes, J. Cottrell, and Y. Bazilevs. Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194(39-41):4135 – 4195, 2005.

[80] D. Jodlbauer, U. Langer, and T. Wick. Parallel block-preconditioned monolithic solvers for fluid-structure interaction problems. *Int. J. Num. Meth. Eng.*, 117(6):623–643, 2019.

[81] D. Jodlbauer and T. Wick. A monolithic fsi solver applied to the fsi 1,2,3 benchmarks. In S. Frei, B. Holm, T. Richter, T. Wick, and H. Yang, editors, *Fluid-Structure Interaction: Modeling, Adaptive Discretization and Solvers*, Radon Series on Computational and Applied Mathematics. Walter de Gruyter, Berlin, 2017.

[82] C. Johnson. *Numerical solution of partial differential equations by the finite element method*. Cambridge University Press, Campridge, 1987.

[83] N. Kikuchi and J. Oden. *Contact problems in elasticity.* Studies in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1988.

[84] D. Kinderlehrer and G. Stampacchia. *An Introduction to Variational Inequalities and Their Applications.* Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 2000.

[85] K. Königsberger. *Analysis 1.* Springer Lehrbuch. Springer, Berlin – Heidelberg – New York, 6. auflage edition, 2004.

[86] K. Königsberger. *Analysis 2.* Springer Lehrbuch. Springer, Berlin – Heidelberg – New York, 5. auflage edition, 2004.

[87] E. Kreyszig. *Introductory functional analysis with applications.* Wiley, 1989.

[88] K. Kumar, M. Wheeler, and T. Wick. Reactive flow and reaction-induced boundary movement in a thin channel. *SIAM J. Sci. Comput.*, 35(6):B1235–B1266, 2013.

[89] O. Ladyzhenskaya. *The boundary value problems of mathematical physics.* Springer, New York, 1985.

[90] R. le Veque. Top ten reasons to not share your code (and why you should anyway). http://faculty.washington.edu/rjl/pubs/topten/topten.pdf, Dec 2012.

[91] R. Liu. *Discontinuous Galerkin Finite Element Solution for Poromechanics.* PhD thesis, The University of Texas at Austin, 2004.

[92] R. Liu, M. Wheeler, C. Dawson, and R. Dean. A fast convergent rate preserving discontinuous galerkin framework for rate-independent plasticity problems. *Comp. Methods Appl. Mech. Engrg.*, 199:3213–3226, 2010.

[93] M. Luskin and R. Rannacher. On the soothing property of the Crank-Nicolson scheme. *Applicable Analysis*, 14(2):117 – 135, 1982.

[94] A. Mikelić, M. Wheeler, and T. Wick. A phase-field approach to the fluid filled fracture surrounded by a poroelastic medium. ICES Report 13-15, Jun 2013.

[95] A. Mikelić and M. F. Wheeler. On the interface law between a deformable porous medium containing a viscous fluid and an elastic body. *M3AS*, 22(11):32, 2012.

[96] A. Mikelić, M. F. Wheeler, and T. Wick. A quasi-static phase-field approach to pressurized fractures. *Nonlinearity*, 28(5):1371–1399, 2015.

[97] J. Nocedal and S. J. Wright. *Numerical optimization.* Springer Ser. Oper. Res. Financial Engrg., 2006.

[98] A. Quarteroni, F. Saleri, and P. Gervasio. *Scientific computing with MATLAB and Octave.* Texts in computational science and engineering. Springer, 2014.

[99] R. Rannacher. Finite element solution of diffusion problems with irregular data. *Numer. Math.*, 43:309–327, 1984.

[100] R. Rannacher. On the stabilization of the Crank-Nicolson scheme for long time calculations. Preprint, August 1986.

[101] R. Rannacher. Finite element methods for the incompressible Navier-Stokes equations. Lecture notes, August 1999.

[102] R. Rannacher. Numerische methoden der Kontinuumsmechanik (Numerische Mathematik 3). Vorlesungsskriptum, 2001.

[103] R. Rannacher. Numerische methoden für gewöhnliche Differentialgleichungen (Numerische Mathematik 1). Vorlesungsskriptum, 2002.

[104] R. Rannacher. Numerische methoden für partielle Differentialgleichungen (Numerische Mathematik 2). Vorlesungsskriptum, 2006.

[105] R. Rannacher. Analysis 2. Vorlesungsskriptum, 2010.

[106] R. Rannacher. *Numerik partieller Differentialgleichungen.* Heidelberg University Publishing, 2017.

[107] R. Rannacher and J. Vihharev. Adaptive finite element analysis of nonlinear problems: balancing of discretization and iteration errors. *Journal of Numerical Mathematics*, 21(1):23–61, 2013.

[108] H.-J. Reinhardt. Die methode der finiten elemente. Vorlesungsskriptum, Universität Siegen, 1996.

[109] S. I. Repin and S. A. Sauter. *Accuracy of Mathematical Models: Dimension Reduction, Homogenization, and Simplification.* European Mathematical Society, 2020.

[110] J. Rice. *Mathematical analysis in the mechanics of fracture*, pages 191–311. Academic Press New York, chapter 3 of fracture: an advanced treatise edition, 1968.

[111] T. Richter. Numerische methoden für gewöhnliche und partielle differentialgleichungen. Lecture notes, Heidelberg University, 2011.

[112] T. Richter. *Fluid-structure interactions: models, analysis, and finite elements.* Springer, 2017.

[113] T. Richter and T. Wick. Variational localizations of the dual weighted residual estimator. *Journal of Computational and Applied Mathematics*, 279(0):192 – 208, 2015.

[114] T. Richter and T. Wick. *Einführung in die numerische Mathematik - Begriffe, Konzepte und zahlreiche Anwendungsbeispiele.* Springer, 2017.

[115] Y. Saad. *Iterative methods for sparse linear systems.* SIAM, 2003.

[116] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3), 1986.

[117] L. Samolik and T. Wick. Numerische Mathematik II (Numerik GDGL, Eigenwerte). Institute for Applied Mathematics, Leibniz Universität Hannover, Germany, 2019.

[118] M. Schäfer and S. Turek. *Flow Simulation with High-Performance Computer II*, volume 52 of *Notes on Numerical Fluid Mechanics*, chapter Benchmark Computations of laminar flow around a cylinder. Vieweg, Braunschweig Wiesbaden, 1996.

[119] M. Schmich and B. Vexler. Adaptivity with dynamic meshes for space-time finite element discretizations of parabolic equations. *SIAM J. Sci. Comput.*, 30(1):369 – 393, 2008.

[120] H. R. Schwarz. *Methode der finiten Elemente.* Teubner Studienbuecher Mathematik, 1989.

[121] R. C. Smith. *Uncertainty Quantification: Theory, Implementation, and Applications.* SIAM, 2014.

[122] M. Steinbach, J. Thiele, and T. Wick. Algorithmisches programmieren (numerische algorithmen mit c++). Hannover : Institutionelles Repositorium der Leibniz Universität Hannover, DOI: https://doi.org/10.15488/11583, December 2021.

[123] F. Suttmeier. *Numerical solution of Variational Inequalities by Adaptive Finite Elements.* Vieweg+Teubner, 2008.

[124] F. T. Suttmeier and T. Wick. Numerische Methoden für partielle Differentialgleichungen (in german). Notes (Vorlesungsmitschrift) at University of Siegen, 2006.

[125] R. Temam. *Navier-Stokes Equations: Theory and Numerical Analysis.* AMS Chelsea Publishing, Providence, Rhode Island, 2001.

[126] T. Tezduyar. Finite element methods for flow problems with moving boundaries and interfaces. *Archives of Computational Methods in Engineering*, 8(2):83–130, 2001.

[127] J.-P. Thiele. Development and test of a nesting method for an embedded lagrangian particle model. Master's thesis, Leibniz University Hannover, Sep 2017.

[128] I. Toulopoulos and T. Wick. Numerical methods for power-law diffusion problems. *SIAM Journal on Scientific Computing*, 39(3):A681–A710, 2017.

[129] R. Trémolières, J. Lions, and R. Glowinski. *Analyse numérique des inéquations variationnelles*. Dunod, 1976.

[130] R. Trémolières, J. Lions, and R. Glowinski. *Numerical Analysis of Variational Inequalities*. Studies in Mathematics and its Applications. Elsevier Science, 2011.

[131] F. Tröltzsch. *Optimale Steuerung partieller Differentialgleichungen - Theorie, Verfahren und Anwendungen*. Vieweg und Teubner, Wiesbaden, 2nd edition, 2009.

[132] S. Turek. *Efficient solvers for incompressible flow problems*. Springer-Verlag, 1999.

[133] S. Turek, L. Rivkind, J. Hron, and R. Glowinski. Numerical analysis of a new time-stepping θ-scheme for incompressible flow simulations. Technical report, TU Dortmund and University of Houston, 2005. Dedicated to David Gottlieb on the occasion of his 60th anniversary.

[134] D. Werner. *Funktionalanalysis*. Springer, 2004.

[135] T. Wick. *Adaptive Finite Element Simulation of Fluid-Structure Interaction with Application to Heart-Valve Dynamics*. PhD thesis, University of Heidelberg, 2011.

[136] T. Wick. Fluid-structure interactions using different mesh motion techniques. *Computers and Structures*, 89(13-14):1456–1467, 2011.

[137] T. Wick. Solving monolithic fluid-structure interaction problems in arbitrary Lagrangian Eulerian coordinates with the deal.II library. *Archive of Numerical Software*, 1:1–19, 2013.

[138] T. Wick. Modeling, discretization, optimization, and simulation of fluid-structure interaction. Lecture notes at Heidelberg University, TU Munich, and JKU Linz available on https://www-m17.ma.tum.de/Lehrstuhl/LehreSoSe15NMFSIEn, 2015.

[139] T. Wick. Modified Newton methods for solving fully monolithic phase-field quasi-static brittle fracture propagation. *Computer Methods in Applied Mechanics and Engineering*, 325:577 – 611, 2017.

[140] T. Wick. *Variational-Monolithic ALE Fluid-Structure Interaction: Comparison of Computational Cost and Mesh Regularity Using Different Mesh Motion Techniques*, pages 261–275. Springer International Publishing, Cham, 2017.

[141] T. Wick. *Multiphysics Phase-Field Fracture: Modeling, Adaptive Discretizations, and Solvers*. De Gruyter, Berlin, Boston, 2020.

[142] T. Wick. Sergey i. repin, stefan a. sauter: 'accuracy of mathematical models'. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 2020.

[143] T. Wick and P. Bastian. Pec3 spring school on introduction to numerical modeling with differential equations. Hannover : Institutionelles Repositorium der Leibniz Universität Hannover, 2019. DOI: https://doi.org/10.15488/6427, 2019.

[144] J. Wloka. *Partielle Differentialgleichungen*. B. G. Teubner Verlag, Stuttgart, 1982.

[145] J. Wloka. *Partial differential equations*. Cambridge University Press, 1987.

[146] E. Zeidler. *Nonlinear functional analysis - nonlinear monotone operators*. Springer-Verlag, 1990.

Index