Karlsruher Institut für Technologie
Doktorarbeit

A Fully Parallelized and Budgeted Multi-level Monte Carlo Framework for Partial Differential Equations

From Mathematical Theory to Automated Large-Scale Computations

Zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

von der KIT-Fakultät für Mathematik des
Karlsruher Instituts für Technologie (KIT)
genehmigte

Dissertation

von
Niklas Lucian Florentin Baumgarten

Tag der mündlichen Prüfung: 08.02.2023

1. Referent: Prof. Dr. Christian Wieners
2. Referent: Prof. Dr. Tobias Jahnke

Karlsruher Institut für Technologie

Abstract

All collected data on any physical, technical or economical process is subject to uncertainty. By incorporating this uncertainty in the model and propagating it through the system, this data error can be controlled. This makes the predictions of the system more trustworthy and reliable. The multi-level Monte Carlo (MLMC) method has proven to be an effective uncertainty quantification tool, requiring little knowledge about the problem while being highly performant.

In this doctoral thesis we analyse, implement, develop and apply the MLMC method to partial differential equations (PDEs) subject to high-dimensional random input data. We set up a unified framework based on the software M++ to approximate solutions to elliptic and hyperbolic PDEs with a large selection of finite element methods. We combine this setup with a new variant of the MLMC method. In particular, we propose a budgeted MLMC (BMLMC) method which is capable to optimally invest reserved computing resources in order to minimize the model error while exhausting a given computational budget. This is achieved by developing a new parallelism based on a single distributed data structure, employing ideas of the continuation MLMC method and utilizing dynamic programming techniques. The final method is theoretically motivated, analyzed, and numerically well-tested in an automated benchmarking workflow for highly challenging problems like the approximation of wave equations in randomized media.

Contents

1 Motivation and Introduction

All models are wrong, but some are useful - George Box, 1978, [41]

The process of model and hypothesis building and their falsification is central to the scientific method of any discipline. In particular, in natural sciences, engineering and economic studies these models are often mathematical systems giving an incredibly precise and well-defined description of the real-world problem. However, even if the mathematical model does account for all aspects of the underlying system, the significance of the predictions is vulnerable to uncertainty in the input data. Picking up on the quote by George Box, we raise the following questions:

- How wrong is the model and can we quantify the error?
- Can we improve the model to better approximate the underlying system?
- Can we measure and discriminate between good and bad models?
- When is a model actually useful and what makes it useful?

Up until the 1990s, uncertainty quantification (UQ) was mostly studied with statistical methods. It became a highly interdisciplinary field by incorporating techniques from numerical analysis, scientific computing and problem domains such as quantum physics or mechanical engineering. Additionally, as solutions to interesting problems in application are usually intractable for human beings, computers and software development are essential within *end-to-end* UQ. In *end-to-end* UQ, we try to consider all contributions to the total error which can be informally expressed as

$$\mathrm{err}_{\mathrm{total}} = \mathrm{err}_{\mathrm{input}} + \mathrm{err}_{\mathrm{model}} + \mathrm{err}_{\mathrm{disc}} + \mathrm{err}_{\mathrm{solve}} + \mathrm{err}_{\mathrm{float}} + \mathrm{err}_{\mathrm{bug}} + \dots . \tag{1.1}$$

It is this sum, which motivates all aspects considered in this thesis.

The input error $\mathrm{err}_{\mathrm{input}}$ stems from incomplete or inaccurate knowledge and can be of various types - uncertain boundary or initial conditions, unknown shapes of the computational domain and unknown material parameters or external forces. Usually, this input data is collected by a measuring process and inherently contains uncertainty and thus, is *aleatoric*. Practical examples for such an *aleatoric* and inherent uncertainty are manufacturing tolerances, weather conditions, noisy communication channels or Heisenberg's uncertainty principle in quantum mechanics. Consequently, as soon as the mathematical model is fed with this uncertain input data, this error has to be propagated through the system.

On the other hand, the model error $\mathrm{err}_{\mathrm{model}}$ might be a result from ignorance, namely that we simply did not consider a certain term or force contributing to the overall system behavior. Thereby, we speak of *unknown* unknowns, which are typically associated with *epistemic* errors due to the lack of knowledge. Often, however, it might even be a conscious decision to introduce this error since a complete and detailed description of the model can result in extremely high computational costs or comes with a sacrifice of mathematical properties of the formulated system. It is in the scientist's responsibility to state and to argue for the invested assumptions and model reductions, as well as to quantify the impact of them. This makes the scientific findings trustworthy and reliable and it acknowledges that high impact decisions might be based on them.

Controlling the discretization error $\mathrm{err}_{\mathrm{disc}}$ has been the main objective for researchers in numerical analysis. In this field, error bounds have been derived, among many others, on quadrature formulas to approximate integrals, time integration schemes to approximate systems evolving over time and finite difference (FD) and finite element methods (FEM) to approximate ordinary and partial differential equations (ODE and PDE). These discretization schemes have, alongside with their implementation on computer systems, given us an incredible leap in understanding nature by simulating its behavior.

With the use of the above discretization methods, finite dimensional algebraic systems arise. Numerical methods to solve these algebraic system introduce an error $\mathrm{err}_{\mathrm{solve}}$, too. As an example, we can think of a Krylov subspace method to solve huge linear systems. In practice, this solver never spans the whole solution space, but approximates the real solution in a subspace up to an acceptable error.

With the use of computers, rounding errors $\mathrm{err}_{\mathrm{float}}$ and cancellation phenomena in floating point arithmetics also have to be considered. In fact, numerical instabilities due to floating point operations still lead to special requirements in high performance computing and in the algorithmic design.

Lastly, programming errors (commonly called *bugs*) become more likely with an increasing stack of involved algorithms to control the other errors. They are also of *epistemic* nature since we are not aware of their existence while executing the program. The only hope to actually control this error is to separate the responsibilities in the software and to continuously and automatically verify the correctness of the code as changes are made to it. In the sum (1.1) further error terms might have to be added. However, the intent here was to list the main contributors considered in this thesis.

To summarize, there is no complete information in real life and there is no perfect algorithm delivering exact results. However, the need for model building and error quantification is already given by the simple fact that it does decrease the cost and development time of new designs and scientific findings dramatically. Moreover, with global crises like climate change, pandemics or supply chain disruptions and rising prices, we heavily rely on risk quantification in order to make the right decisions. For a more detailed motivation on UQ, we refer to the books by Sullivan [241], Soize [229] or McClarren [180] and the reference section.

1.1 On Computational Methods

Computational methods and simulations are used in almost every scientific discipline, having the main goal to predict the behavior of the underlying system they try to replicate. By doing so, they can get notoriously complicated, such that even on the fastest and biggest supercomputers no solution or approximation with an acceptable accuracy can be found. This raises the demand for sophisticated and ever-evolving numerical and uncertainty quantification methods. In the following, we distinguish between these numerical methods like FEMs to find approximative solutions to PDEs and UQ methods to control the uncertainty entering the system by the input data.

The use of FEMs to solve PDEs has proven to be greatly useful, as they come with rigorous convergence results for many applications. Beyond this, they are deployable on parallel computing clusters and have been implemented in many commercial and non-commercial open source software tools. Even though FEMs are well-established, new methods still arise, tailor-made for specific model problems in order to cope with low regularity, high computational demands, multi-physics systems, stability issues or missing conservation of certain physical quantities. We refer to standard literature like the books by Brenner and Scott [44], Ern [81] and Braess [42] to underline the importance and wide range of applications of FEMs. Furthermore, we also mention more recent developments in the field of mixed FEMs, put together in the book by Boffi, Brezzi and Fortin in [38] or again by Ern [70] about discontinuous Galerkin (DG) methods. However, up until the early 1990s uncertainty in the input data was not taken into account. Early attempts to tackle this with Monte Carlo (MC) methods are for example described in [36] and [162].

To capture briefly the history of MC methods, we recall a very early application of the method in the estimation of π with a rather simple experimental setup. We consider a unit square and draw a quarter of a circle with radius one around the origin in the lower left corner. Then, we sample from a uniformly distributed random variable on the unit square and compute the distance of each sample to the origin. Lastly, we take the ratio from the amount of samples with a distance smaller than one to the total sample amount which gives in the limit $\pi/4$. Surely, there are way better ways to compute π as the convergence rate of the MC method is rather slow with $\mathcal{O}(M^{-1/2})$, M being the total number of samples. Nevertheless, Enrico Fermi, Stanislaw Ulam, John von Neumann and Nicholas Metropolis found soon enough other applications for the MC method in nuclear and particle physics [181] as well as in military research, most famously in the Manhattan project [127]. The MC method requires little assumptions on the underlying system and is extremely simple to implement. Thereby the MC method, and the various variants of it, have been used in applications in biology, chemistry, computer science, economics and finance, physics and engineering. We refer to the book by Liu [168] discussing many applications and strategies for MC methods.

The MC method belongs to the class of non-intrusive statistical estimators, that is, the method is based on independent sample-wise experiments. Within the scope of this thesis such an "experiment" is essentially the task of finding an approximative solution to a PDE. This is by itself already a computationally challenging task. Therefore, the great benefit of such non-intrusive methods is that they can build on existing tools to solve sample-wise the deterministic problems. Additionally, as the used samples are required to be independently and identically distributed (iid), the methods provide a natural parallelization scheme and are deployable on extremely large computing clusters. Nevertheless, the poor convergence rate of $\mathcal{O}(M^{-1/2})$ leads to the evaluation of a huge amount of samples M to achieve an acceptable accuracy, which has motivated numerous authors to find better methods.

The Quasi-Monte Carlo (QMC) method is as the MC method a quadrature rule to an expectation integral, but where the MC method chooses its quadrature points (pseudo-)randomly, the QMC method employs a quasi-random selection instead. Early works on this method go back to Niederreiter [198, 199] as well as Sloan and co-authors [226, 71] and Calfisch [51]. It finds applications in financial product valuation [204, 106] and if the problem satisfies certain conditions, the convergence rate gets close to $\mathcal{O}(M^{-1})$ (confer [10] or the other mentioned resources).

As the QMC method, the stochastic collocation (SC) method also employs a deterministic choice of quadrature (collocation) points [265]. However, the weights of the quadrature rule are not uniform as in the MC or QMC method. The collocation points are taken from a sparse tensor grid [227, 48] and if the solution provides sufficient regularity with respect to the stochastic parameter, this results in very fast convergence. SC methods have for example been used by Xiu to solve differential equations with random input [264].

The stochastic Galerkin (SG) method does differ from the other methods substantially as it is not a non-intrusive method. It aims to satisfy the governing equations in a weak form and not at given sample points by incorporating the uncertainty in the ansatz space to the weak formulation. In particular, the solution is sought in a polynomial basis which usually consists of orthonormal polynomials with respect to the probability distribution function. Hence, the solution is expressed in an expansion which is commonly called generalized polynomial chaos (gPC) expansion. This expansion is then plugged into the PDE which then is solved with a Galerkin approximation. Further details can be found in the upcoming section or in the book by Ghanem [104].

Multi-level Monte Carlo (MLMC) methods have been receiving much attention since the initial publication by Heinrich [135], who was inspired by a multigrid approach for fluids under gravity [43]. Giles [109, 110] applied the MLMC method in mathematical finance to stochastic differential equations (SDEs). Since then, he has been continuously contributing to the field [108] and maintains a list of all publications on MLMC methods on [107]. The core benefit of the MLMC algorithm is that it finds a way to balance the computational cost of samples on different discretization levels and the total number of samples, resulting in an efficient algorithm. To refer to equation (1.1), the algorithm basically invests computational resources in controlling $\mathrm{err}_{\mathrm{input}}$ and $\mathrm{err}_{\mathrm{disc}}$. This makes it very attractive for simulations where the model evaluation of one sample is exceptionally expensive.

All the mentioned methods face common challenges one way or the other. For example, a high dimensional stochastic space results in worse performance of all methods. Whereas for the MC-type methods the convergence rate is independent of the stochastic dimension, the variance, and thus the constant in the convergence result, often is not [92]. However, for SC-type methods a large dimension can result in a dramatic explosion of computational effort. There is no easy answer to which method is actually faster or even applicable, as the answer to this question is highly problem dependent. Therefore, knowledge from the application domain is often of high importance to actually design the best methods, and beyond this, it is also needed in order to root out all epistemic errors.

Goals of the Method Design

Within this thesis, we think of a method as a combination of algorithms to solve mathematical problems with soft- and hardware systems to control the total error in (1.1). The complexity of this problem requires to tackle several issues at once and thereby to consider the above definition of a method. For epistemic errors, e.g. resulting from model simplification or software bugs, this is really hard to rigorously quantify in a mathematical way. Nevertheless, if we impose to the method that it should deliver fast feedback about its validity and its performance on a particular problem, we can get back control over epistemic errors in some way. Fast feedback and a method design from the application point of view is crucial to be actually able to distinguish between good and bad models and to incorporate knowledge from the problem domain.

For the parts of the error (1.1) where we actually have convergence results, we want to achieve a balance in the invested resources. We know by utilizing more computational resources that the discretization error $\mathrm{err}_{\mathrm{disc}}$ gets smaller. By having a suitable stochastic model for the input data we also know that, by computing more samples, the input error $\mathrm{err}_{\mathrm{input}}$ is controlled as well. Thereby, to be actually able to tell which algorithm stack is indeed the best one for a particular problem, we want to be able to answer the question, which method leads to the smallest total error (1.1), given a certain computational budget. Unfortunately, in most cases we can only prove and numerically verify the convergence of an error with respect to some discretization parameter or consumed computational resources, not the actual size of the error. Nevertheless, this asymptotic behavior and a-posteriori error estimates are often the only results that can be achieved and are thereby good enough for our goal.

Lastly, we want the developed methods to be robust. To define robustness, we want the method to deliver fast and reliable results under various circumstances which might be a varying problem class, an interplay of different algorithms or different soft- and hardware architectures. The overall method is exposed to a wide range of influences of theoretical or technological nature, and we demand that the approach is robustly performing under all of them.

Summed up, we want to develop robust methods yielding repeatable and reliable results, where the methodology does account for all aspects of the error equation (1.1), invests computational resources economically and which gives fast feedback for the model building process. Thereby, we aim for a holistic method design - *from mathematical theory to automated large-scale computations.*

1.2 Main Contributions and Outline

In order to achieve the outlined goal of a holistic method design to approximate solutions of PDEs subject to uncertain input data, we draw the attention to the Venn diagram 1.1. To find the approximative solutions, we have mostly focused our work on three main areas.

First, we aimed for the unified application of various finite element methods on a new parallel data structure

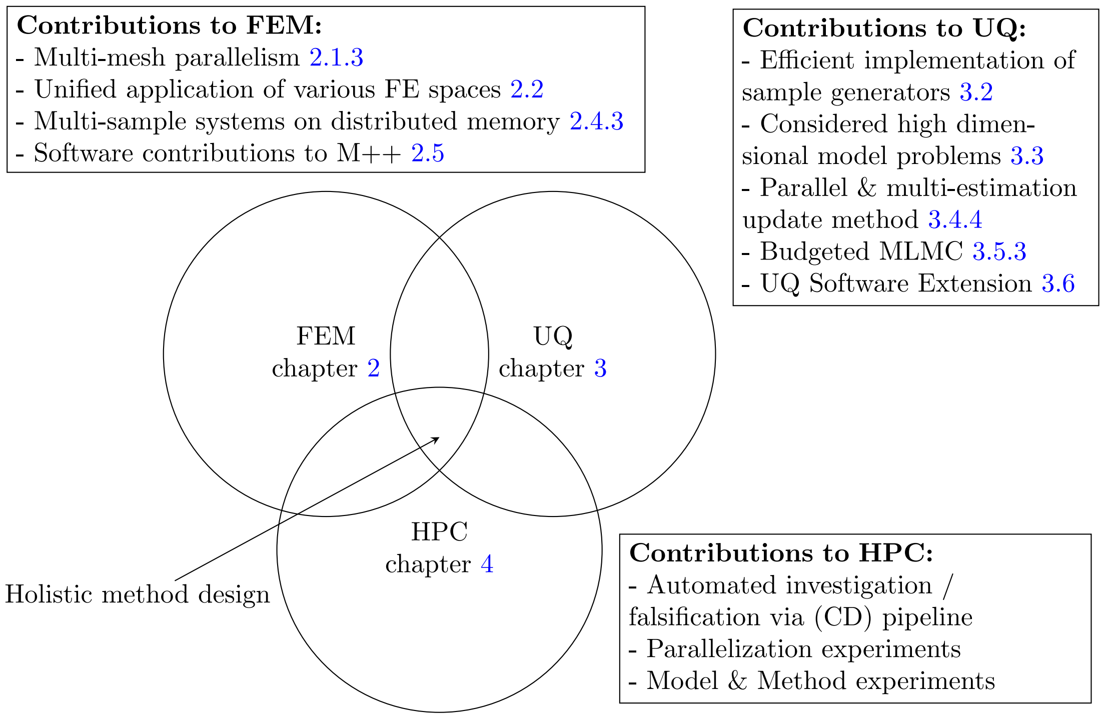

Contributions to FEM:
- Multi-mesh parallelism 2.1.3
- Unified application of various FE spaces 2.2
- Multi-sample systems on distributed memory 2.4.3
- Software contributions to M++ 2.5
Contributions to UQ:
- Efficient implementation of sample generators 3.2
- Considered high dimensional model problems 3.3
- Parallel & multi-estimation update method 3.4.4
- Budgeted MLMC 3.5.3
- UQ Software Extension 3.6
FEM chapter 2
UQ chapter 3
HPC chapter 4
Holistic method design
Contributions to HPC:
- Automated investigation / falsification via (CD) pipeline
- Parallelization experiments
- Model & Method experiments

Figure 1.1: Main contributions of this thesis in a Venn diagram.

suitable for UQ problems. To do so, we used and extended the parallel FEM library M++. The intent behind this is to develop a framework with which PDEs of different types can be numerically approximated. This includes the approximation of highly challenging problems like wave equations with randomized wave speeds or transport equations on randomized flux fields in multiple space dimensions (confer section 1.3 for the problem definition). To do so, we use space-time discontinuous Galerkin approximations and implicit time-stepping methods to discretize the deterministic version of the PDE. Furthermore, we have employed many other discretizations like the enriched Galerkin discretization or mixed FEM for elliptic model problems and utilized the broad selection of numerical solvers M++ [28] is offering. A central contribution to M++ is the new multi-mesh parallelization which can be employed for arbitrary finite element spaces and non-intrusive UQ algorithms. This parallelization follows a very simple, yet extremely adaptive rule for the load distribution. Shortly speaking, the idea is to assemble huge linear systems on distributed memory over multiple computing nodes, while the structure of the system is adapted to the needs of the problem. This integrated approach intertwines the FEM with the UQ methods resulting in a highly efficient and hardware proximal implementation without sacrificing the non-intrusiveness of the UQ methods.

Secondly, we have designed the new parallel data structure such that any non-intrusive UQ method can easily be executed on it, i.e., we used it for Monte-Carlo-like and stochastic collocation methods. In particular, we employed it in a new method we propose with this thesis - the budgeted MLMC (BMLMC) method. The BMLMC method breaks up the initial problem into multiple subsequent problems, where each is solved under an optimality condition. Thereby, the BMLMC method utilizes techniques from dynamic programming and requires basically no a priori knowledge about the problem or its computational complexity. This method is motivated by the fact that in high performance computing applications a computational budget has to be reserved in order to deploy a job on the computing cluster. With the BMLMC method this can be done by imposing the reserved computational budget as a constraint to the problem, i.e., the method solves a knapsack problem. The method inherits the properties of the MLMC method and has proven to be extremely robust, highly performant, and widely applicable in combination with the new parallelization. Lastly, we want to mention that with this method we can empirically study the question: Which algorithm stack should be used to minimize (1.1)? This is done in particular by imposing the same computational budget to all algorithm combinations and deciding for the one which gives the smallest error estimate.

Lastly, we mention that these empirical investigations are done completely automated. In order to do so we have developed a continuous delivery (CD) pipeline which builds and verifies the software prior to the deployment on the high performance computing cluster. Then, the newest version of the software is executed as a benchmark job offering fast feedback about the method development and the model building process. This workflow enables stable, sustainable and transparent software engineering, continuously checked and benchmarked, such that further work can easily and confidently build up on the achieved results.

Concluded, one main contribution is the unified application of a broad selection of finite element and uncertainty quantification methods in a single framework. Beyond this, we propose the new BMLMC method and the new parallel data structure based on multi-sample systems, as well as the automated investigation with a CD pipeline. Lastly, we want to emphasize that a central research product of this thesis is the piece of software handling all the above.

Outline

We will close this introductory chapter by giving a detailed study on recent literature on uncertainty quantification and scientific computing in section 1.3. We will focus on publications which are related to the topics and problems we want to investigate in this thesis.

By drawing the attention to diagram 1.1 one more time, we also want to motivate the structure of the main chapters of this thesis. Namely, chapter 2 introduces the theory on FEM, chapter 3 explains UQ applications and algorithms and chapter 4 discusses the numerical results achieved on the HoReKa super computer [249]. The actual outline within chapter 2 and 3 is further motivated by the build structure of the developed software, i.e., we will introduce the mathematical theory to the reader in the same order as we define the objects for the C++ compiler. In particular, this means for the FEM chapter that we explain the triangulation of the computational domains in 2.1 and the multi-mesh parallelism. This section is followed by an overview on the used finite element spaces in 2.2 and the discretized deterministic model problems in 2.3. In 2.4, we shortly explain the usage of the numerical solvers on multi-sample systems before we close the chapter with section 2.5, giving an overview on the project and software structure of M++.

The next chapter on UQ starts by recalling some theory on random fields and Bochner spaces in 3.1, and based on that theory, we explain how we efficiently sample from the input data in 3.2 to hand it over to the PDE solver. We will further investigate the uncertain model problems in 3.3 and derive some of their properties. Towards the end of the chapter, we explain the used UQ algorithms in the sections 3.4 and 3.5, where we will also explain the development of the BMLMC method in full detail and analyze its properties. Lastly, we present a discussion on the parallelization properties of the overall method and finish the chapter with a short recapitulation of the UQ extension to M++ in 3.6.

The last main chapter 4 of this thesis outlines the numerical results achieved on the HoReKa supercomputer in a unified, automated and large-scale (on up to 2048 processing units) manner. We will present model building experiments, algorithmic benchmarks, investigations on the scaling of the parallelization as well as other parameters which can influence the outcome of a computation. In particular, we will consider elliptic and hyperbolic model problems subject to different uncertain input data, approximated with various discretizations and solved with highly performant algorithms.

Publications and Software Releases

Parts of the work here presented have been subject to publications and software releases. The foundational work to the framework was published in [28] where we considered elliptic, parabolic and hyperbolic model problems to demonstrate the non-intrusiveness and the abstract applicability of the framework. However, even though the overarching theory remained the same, the developed software has changed substantially since this publication. We want to refer to the release notes {1} for a high-level documentation of the changes. Following the best practice of short release cycles as suggest in most software engineering workflows, we have been contributing to 26 stable releases of M++ of which 3 are published on *radar4kit* [258, 259, 260] and to 4 stable releases of MLUQ – the UQ extension for M++.

1.3 Existing Research and Model Problems

In the following, we present the model problems used within this thesis, as well as existing literature investigating them with different methods. The intent of this section is to give a first impression of the applications and to identify existing results. We consider for all model problems that they are defined on a bounded polygonal domain $D \subset \mathbb{R}^{\mathbf{d}}$ with $\mathbf{d} \in \{1, 2, 3\}$, a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ and possibly a time interval $[0, T] \subset \mathbb{R}$. We denote the outer unit normal vector on the boundary ∂D with $\mathbf{n}$ and subsets of the boundary with $\Gamma \subset \partial D$. For now, we assume that the input data is imposed to the model problems, such that they admit randomly distributed solutions. We are interested in the solution map $\Omega \to V$, $\omega \mapsto u(\omega)$, where V is the solution space of a deterministic PDE. However, in many applications, we are not able to obtain this complicated mathematical object. To develop an intuition to the solution map $\Omega \to V$, we instead denote for each $\omega^{(m)} \in \Omega$ a sample of the input data with $\mathbf{y}^{(m)}$ and a sample of the solution with $u^{(m)} \in V$. Then we can compute statistical moments of a quantity of interest (QoI) of the solution, which in most cases is a bounded functional $\mathrm{Q} \colon V \to \mathbb{R}$, such that for some constant $C > 0$

$$|\mathrm{Q}(u) - \mathrm{Q}(v)| \leq C \left\| u - v \right\|_V , \quad \forall u, v \in V \tag{1.2}$$

QMC	SC	MLMC	CMLMC	MLQMC	MLSC	MIMC	MIQMC	MISC
[118, 217, 120, 105]	[12, 200, 201, 13, 35, 89]	[25, 54, 58, 55, 246, 1, 53, 130, 153, 144]	[61]	[157]	[133, 262, 245, 132, 160]	[128]	[214]	[129]

Table 1.1: Literature which considers elliptic model problems and non-intrusive UQ. For an overview on the used abbreviations we refer to table 1.5 in the upcoming section.

holds. Suitable QoI will be given in the following for each model problem. To summarize, we are interested in approximations to the expectation value $\mathbb{E}[\mathrm{Q}(u(\omega))]$ or higher statistical moments of the *forward* problem, i.e., given a distribution of the input data, we address the question: what are the statistical moments of QoIs to the randomly distributed solutions of the system.

Subsurface Diffusion Problem

We start by introducing the prototypical model problem in UQ and scientific computing - an elliptic partial differential equation (PDE) subject to random input data. This PDE is derived by Darcy's law and is often used to model subsurface flows in random (unknown) porous media (see for example in [195, 196, 67] or the reference section). The random input data is often employed to capture the uncertainties resulting from finitely many measurements of the permeability in the ground. Besides application in geophysics, similar PDEs arise in many engineering applications. This model problem is of particular interest in UQ since the (deterministic) theory is well-established and many legacy software tools exist to solve it numerically. This enables an efficient method development as solutions to this equation typically provide sufficient regularity with respect to the input data. The elliptic model problem reads as follows.

Model Problem 1.1. We search for a solution $u\colon \Omega \times \overline{D} \to \mathbb{R}$, such that

$$\begin{cases} -\operatorname{div}(\kappa(\omega,\mathbf{x})\nabla u(\omega,\mathbf{x})) &= f(\omega,\mathbf{x}) & \mathbf{x} \in D \\ \kappa(\omega,\mathbf{x})\nabla u(\omega,\mathbf{x}) \cdot \mathbf{n} &= g_{\mathrm{N}}(\omega,\mathbf{x}) & \mathbf{x} \in \Gamma_{\mathrm{N}} \\ u(\omega,\mathbf{x}) &= u_{\mathrm{D}}(\omega,\mathbf{x}) & \mathbf{x} \in \Gamma_{\mathrm{D}} \end{cases}$$

where the permeability $\kappa\colon \Omega \times \overline{D} \to \mathbb{R}$, the source term $f\colon \Omega \times D \to \mathbb{R}$, the Dirichlet boundary data $u_{\mathrm{D}}\colon \Omega \times \Gamma_{\mathrm{D}} \to \mathbb{R}$ and the Neumann boundary data $g_{\mathrm{N}}\colon \Omega \times \Gamma_{\mathrm{N}} \to \mathbb{R}$ are represented by random fields. The subsurface flux can be computed afterwards via $\mathbf{q}(\omega,\mathbf{x}) = -\kappa(\omega,\mathbf{x})\nabla u(\omega,\mathbf{x})$.

Common models for the random permeability are log-normal fields with exponential covariance (confer section 3.1.2). Typical QoIs are the L^2- or the H^1-norm of the solution or measures of the outflow of the domain which might be of more practical interest. Over the last ten years, this problem has received much attention which led to a rapid development of methods and further specifications of the problem. We refer to table 1.1 for an overview.

Other than the standard Monte Carlo (MC) method to estimate the expectation value $\mathbb{E}[\mathrm{Q}]$ (e.g. as explained in [268] for this particular problem), many new UQ methods have been developed in recent years for elliptic PDEs in order to improve on the convergence rate of $\mathcal{O}(M^{-1/2})$, M being the amount of samples.

In [118] Graham, Kuo, Scheichl and Sloan propose a QMC method for this elliptic PDE a with log-normally distributed $\kappa(\omega, \mathbf{x})$. The samples are created with the circulant embedding algorithm, first published in [72], and the PDE is solved with a mixed FEM (a detailed description of this method can be found in [46]). This work is then later extended in [120] with a more detailed analysis to the combination of QMC and circulant embedding methods.

This elliptic model problem has also served as an early application case for SC methods, most famously in the work by Babuška, Nobile and Tempone in [12]. This work has later been extended in [200, 201, 13] and exploits the *finite dimensional noise assumption* (confer assumption 3.15) in order to replace the stochastic space by a parametric space. This ansatz is under development up until recently, for example in [89] and [224], where adaptive variants of this method are proposed.

The MLMC method also found very early applications to this problem for example by two research groups around Scheichl in [58] and Schwab [25]. This work is proceeded in [55, 53] and [246] where a very detailed error analysis is presented. In [1], the method is combined with multiscale methods, in [153], the problem class is extended to elliptic variational inequalities. Furthermore, pore-scale rock problems [144], permeabilities modeled as Besov random trees [222] and problem configurations with uncertain boundaries [56] also have been considered. Later works propose new ways to parallelize the MLMC algorithm such as [217, 16]. This will be investigated within this thesis, too, and is based on the work we published in [28]. An algorithmic modification is presented in [61] via a continuation MLMC (CMLMC) method. This method served as inspiration for the BMLMC method which we will propose in this thesis.

The multilevel idea has also spread to QMC methods [157] and SC methods [133, 262, 245, 132, 160] and [145, 231] for other model problems. Furthermore, the idea has been extended to multiple indices as in [128], where different spatial dimensions get assigned their own level. Lastly, we want to mention [214, 129] combining this multi-index approach with SC and QMC.

All the presented sources introduce methods to the *forward* problem. For a collection of literature on other methods which are not applied for forward UQ, we refer to table 1.2. In practice, the *inverse* problem might be of even greater interest. The scenario in this problem is the following: Assume that we have knowledge about the boundary conditions and source terms of model problem 1.1. Furthermore, we can retrieve parts of the solution from measurement data. The task is to find a distribution of the permeability. This problem can be formulated in a Bayesian way, and it is often a rather challenging since it is ill-posed. For an introduction to the Bayesian perspective to inverse problems, we refer to [147, 236].

Regarding the elliptic inverse model problem, there has also been very active research for example by Stuart in [68]. Typically, some sort of Markov Chain Monte Carlo (MCMC) methods are involved in solving these Bayesian inverse problems, which have been combined with the multi-level idea and QMC methods in [73, 217, 74].

This elliptic model problem has also served in the development of intrusive UQ methods - namely the stochastic Galerkin (SG) method. Early work goes back again to Babuška, Tempone, Schwab, Xiu and many co-authors in [14, 93, 266].

Very recent work has considered the elliptic model problem also as an uncertain constraint in optimal control problems [100, 101, 175]. In these publications, stochastic gradient descent (SGD) algorithms are proposed and analyzed for this model problem. New algorithmic applications like the SGD method for

No UQ	Intrusive UQ	Inverse UQ	Optimization under UQ
[149, 9, 81, 42, 42, 134, 242, 70, 46, 152, 38, 173, 84, 232, 24, 163]	[14, 93, 266, 112, 113]	[68, 73, 217, 74, 136]	[101, 175, 124]

Table 1.2: Literature which considers elliptic model problems other than non-intrusive forward UQ.

optimal control problems or MCMC methods for Bayesian inversion require efficient forward UQ. Thereby, these outer-loop algorithms have motivated us to design our forward method as integrated as possible to provide an easy and highly performant access to forward model evaluation.

Lastly, neglecting the uncertainty of the model problem, in [9, 134, 70] discontinuous Galerkin (DG) methods and in [242, 163] enriched Galerkin (EG) methods are analyzed for elliptic problems offering new approaches to conserve the flux in the discretization scheme besides mixed FEM [38, 46].

Concluded, it is fair to say, that there has been an extensive study on this model problem. Nevertheless, it almost serves as laboratory for new methods and research questions, such that we used it within the scope of this thesis as well to investigate and test our own method.

Contaminant Transport

With the subsurface diffusion problem, one can compute randomly distributed subsurface flux fields $\mathbf{q}(\omega, \mathbf{x})$. With this random vector field the transport of contaminants through the ground, say nuclear waste or spilled crude oil, can be investigated. Hence, a QoI is for example the total mass of the contaminant at a given time in a given spatial domain.

Model Problem 1.2. We search for the particle density $u\colon \Omega \times \overline{D} \times [0,T] \to \mathbb{R}$, such that

$$\begin{cases} \partial_t u(\omega,\mathbf{x},t) + \operatorname{div}(\mathbf{q}(\omega,\mathbf{x}) u(\omega,\mathbf{x},t)) &= f(\omega,\mathbf{x},t) & \mathbf{x}\in D,\ t\in(0,T] \\ u(\omega,\mathbf{x},t) &= u_{\text{in}}(\omega,\mathbf{x},t) & \mathbf{x}\in\Gamma_{\text{in}},\ t\in(0,T] \\ u(\omega,\mathbf{x},0) &= u_0(\omega,\mathbf{x}) & \mathbf{x}\in D \end{cases}$$

for some randomly distributed Darcy flux $\mathbf{q}\colon \Omega\times D \to \mathbb{R}^{\mathbf{d}}$, right-hand side $f\colon \Omega\times D\times(0,T]\to\mathbb{R}$, initial data $u_0\colon \Omega\times D\to\mathbb{R}$ and inflow boundary condition $u_{\text{in}}\colon \Omega\times\Gamma_{\text{in}}\times(0,T]\to\mathbb{R}$ on $\Gamma_{\text{in}} = \{\mathbf{x}\in\partial D\colon \mathbf{q}\cdot\mathbf{n} < 0\}$.

This model problem is a hyperbolic PDE and even though this is a very easy and prototypical hyperbolic equation, in general this problem class imposes big challenges as the solutions are of low regularity. In the nonlinear case, the solutions can even form shocks, making the analysis and the numerical discretization extremely challenging [179, 22, 66, 166]. A similar model problem was investigated in [156], where Kumar et al. also considered a random Darcy flux as source of uncertainty treated with a MLMC method.

Typical approximation schemes to this equation formulate a semi-discrete system, where the PDE is first discretized over the spatial domain with DG methods [70, 134, 59]. Afterwards the semi-discrete system is solved with a time integrator, for example with an implicit or explicit Runge-Kutta method [60]. In the explicit case, this can cause stability issues, if the Courant-Friedrichs-Lewy (CFL) condition is violated. To make the matter worse, as this PDE is subject to random input data, the CFL condition is random as well and has to be fulfilled for any sample, potentially in a multi-level framework. In [119], a new approach to this

No UQ	MC	SC	MLMC	CMLMC	MCMC
[59, 60, 70, 134, 143, 221, 163]	[195, 196, 162]	[202]	[230, 156, 119, 171]	[208]	[138]

Table 1.3: Literature which with related problems to the hyperbolic and parabolic transport model.

is presented where samples are only considered on stable discretizations, but still in a multi-level framework. The authors have been able to show that the convergence results of the MLMC method asymptotically remain valid.

If the model problem 1.2 is combined with a diffusion term, the solutions to the problem get smoother, and we classify the PDE as parabolic. A similar problem was for example investigated in [230]. This parabolic model problem admits solutions of more regularity than its hyperbolic counterpart. However, especially for a small diffusion term it is not obvious which discretization is the best since DG [143, 221] or Petrov-Galerkin methods with artificial diffusion [148, section 9.2] can be employed to discretize the PDE on the spatial domain. Without the transport term, EG discretizations have also been employed in [163].

To list the further literature on the UQ side for the model, we refer to table 1.3 and remark that a non-linear parabolic problem is the main model in the book [180]. SC methods have also been applied in [202] and MLMC methods in [171] for purely diffusive transport. Lastly, MCMC methods have been applied for the parabolic PDE without transport term for Bayesian inversion with Gaussian priors [138]. Even though we have conducted experiments on the transport diffusion equation we will omit a detailed analysis of this parabolic PDE and only note that a lot of the techniques shown in this work can be applied similarly.

Acoustic Wave Equation

We want to consider another hyperbolic model problem, namely the acoustic wave equation. This model describes the propagation of an acoustic wave over space and time, and thus finds many applications in physics and engineering. Here we consider the case of subsurface wave-propagation, for example caused by earthquakes or used for seismic measurements. Wave-type phenomena are observed everywhere in nature, e.g. in the form of mechanical or electromagnetic waves, where they serve as a carrier for information in the form of light, sound or radio signals. As part of the collaborative research center (CRC) 1173 *Wave phenomena - analysis and numerics*, the goal with the randomized acoustic model problem depicted below is to combine techniques developed by the UQ community with analytical and numerical achievements on wave equations.

Model Problem 1.3. We search for the randomly distributed velocity field $\mathbf{v}\colon \Omega \times \overline{D} \times [0,T] \to \mathbb{R}^{\mathbf{d}}$ and pressure component $p\colon \Omega \times \overline{D} \times [0,T] \to \mathbb{R}$, such that

$$\begin{cases} \rho(\omega,\mathbf{x})\partial_t \mathbf{v}(\omega,\mathbf{x},t) - \nabla p(\omega,\mathbf{x},t) &= \mathbf{f}(\omega,\mathbf{x},t) & \mathbf{x} \in D,\ t \in (0,T] \\ \partial_t p(\omega,\mathbf{x},t) - \kappa(\omega,\mathbf{x})\,\mathrm{div}\,(\mathbf{v}(\omega,\mathbf{x},t)) &= g(\omega,\mathbf{x},t) & \mathbf{x} \in D,\ t \in (0,T] \\ \mathbf{v}(\omega,\mathbf{x},t)\cdot\mathbf{n} &= 0 & \mathbf{x} \in \partial D,\ t \in [0,T] \\ (\mathbf{v},p)(\omega,\mathbf{x},0) &= (\mathbf{v}_0,p_0)(\omega,\mathbf{x}) & \mathbf{x} \in D \end{cases}$$

No UQ & FWI	QMC	SG	SC	MLMC	MLSC	Intrusive
[141, 97, 75, 82, 77, 23, 33, 269, 207, 39]	[98]	[116, 210]	[243, 190, 191, 172]	[186, 238, 185, 184, 187, 188, 18, 21, 122]	[145, 231]	[116, 158, 92, 103, 182, 87]

Table 1.4: Literature which considers related problems to the acoustic wave model.

where $\mathbf{f}\colon \Omega \times D \times (0,T) \to \mathbb{R}^{\mathbf{d}}$ and $g\colon \Omega \times D \times (0,T) \to \mathbb{R}$ are random right-hand sides and where the material parameters $\rho, \kappa^{-1}\colon \Omega \times D \to \mathbb{R}$ are again modeled as random fields. We further allow for randomly distributed initial data in the velocity component $\mathbf{v}_0\colon \Omega \times D \to \mathbb{R}^{\mathbf{d}}$ and the pressure component $p_0\colon \Omega \times D \to \mathbb{R}$.

As for the transport-type equations, common ways to approximate this system are to discretize first in space and then in time. This approach is commonly called method of lines and was numerically analyzed for this model problem in [141] for DG methods and different time integration schemes. However, classical time integration schemes are, as inherently iterative processes challenging to parallelize [97], and thereby space-time methods have received attention in recent years [75, 82, 77, 23, 33]. In this way of discretizing the PDE, we treat the time as an extra spatial dimension resulting in a $(\mathbf{d}+1)$-dimensional stationary problem. Consequently, huge linear systems have to be solved in this approach, but with the promise that this system can be solved in a fully parallelized manner as well as with adaptive equation assembling [269, 207]. It is highly problem-, application- and hardware-dependent which method is more useful or even applicable and can hardly be discussed purely analytically.

The model problem 1.3 is of high interest in geophysics. In particular the inverse problem, i.e., the search for a reconstruction of the material by observations of the solution to the wave equation finds many applications in seismic imaging. However, this is mathematically a highly challenging problem and small errors in the measurement can change the reconstruction of heterogeneous material drastically. New methods to tackle these challenges are for example proposed in [82] using space-time discretizations and in [39] employing a Newton-CG inverse solver with a DG method and a implicit midpoint rule to solve the forward problem.

Adding uncertainty to this model problem makes the problem even harder to approximate. Nevertheless Mishra, Schwab and Šukys considered in [238, 185] acoustic wave equations with random coefficients and solved them with MLMC, finite volume (FV) and explicit time stepping methods. However, the authors faced the challenge of randomized CFL conditions. Even though these issues can be overcome with estimates on the maximum wave speed and adjusting the discretization parameters, we propose to use implicit time-stepping schemes and space-time discretization despite the increased computational effort in order to stabilize the simulations. Our reason to do so is the fact that the cause for the instability might be very localized but still has to be considered on the whole domain if no adaptivity is used. Beyond this, the mentioned group has published further work on hyperbolic conservation laws treated with MLMC methods, e.g. on the shallow-water equation [186], nonlinear hyperbolic conservation laws [184, 187] and generally conservation laws with random flux functions [188]. Furthermore, we want to refer to [21] where seismic wave propagation is considered, too, and [122] where the authors combined MLMC with locally adapted time steps to overcome the stability issues.

Other than the MLMC method, the QMC method has also been employed to wave equations on heterogeneous random media, for example by Ganesh, Kuo, and Sloan in [98] as well as the SC method by Motamed, Nobile and Tempone in [190, 191] for random wave speeds. We further note the convergence analysis in [243] and an application of SC methods to high-frequency wave propagation [172].

Furthermore, early applications of the SG method to wave equations go back to [116]. However, the stochastic Galerkin method can result in oscillatory solutions and might not preserve hyperbolicity which is why entropy variables are introduced and approximated [158, 92]. Further applications of intrusive methods on hyperbolic equations are for example given in [103, 182] and in [87] for Maxwell systems.

Lastly, we want to mention the surveys and books on recent advances for hyperbolic PDEs [193, 2, 3] and refer again on a table 1.4 collecting the mentioned resources.

Abbreviation	**Meaning**
FE	Finite Element
UQ	Uncertainty Quantification
ST	Space-time
DP	Dynamic Programming
CD	Continuous Delivery
CB	Continuous Benchmarking
SG	Stochastic Galerkin
MC	Monte Carlo
SC	Stochastic Collocation
KL	Karhunen-Loève Expansion
RT	Raviart-Thomas
PDE	Partial Differential Equation
HPC	High Performance Computing
FWI	Full Wave Inversion
FFT	Fast Fourier Transform
FEM	Finite Element Method
CFD	Computational Fluid Dynamics
SGD	Stochastic Gradient Descent
QMC	Quasi-Monte Carlo
MSE	Mean Squared Error
MSS	Multi-sample System
CDF	Cumulative Distribution Function
PDF	Probability Density Function
PMF	Probability Mass Function
MLMC	Multi-level Monte Carlo
MLSC	Multi-level Stochastic Collocation
MIMC	Multi-index Monte Carlo
MISC	Multi-index Stochastic Collocation
FDNA	Finite Dimensional Noise Assumption
CMLMC	Continuation Multi-level Monte Carlo
BMLMC	Budgeted Multi-level Monte Carlo
MLQMC	Multi-level quasi-Monte Carlo
MIQMC	Multi-index quasi-Monte Carlo

Table 1.5: Apprevations frequently used in this thesis.

2 Finite Element Methods

The first Finite Element Methods (FEMs) were developed around the 1950s in the aerospace industry. At the early stage of development in the absence of the massive computing power we enjoy today, these methods only found very specific applications as well as a narrow theoretical background. Driven by the exponential growth of computational capacities described by Moore's Law and fundamental theoretical achievements, they have found countless applications in physics, engineering and financial simulations and have become a main driving force for the progression of numerical approximations of partial differential equations (PDEs).

Despite this development, they are still a contemporary topic in applied mathematical research like numerical analysis and scientific computing, as new methods tailored to specific models emerge, leading to a large variety of FEMs.

This chapter will introduce the different FEMs exploited to approximate the deterministic version of the model problems 1.1-1.3 in space and potentially in time. As a start, we describe the decomposition of the computational domain into a set of finite subdomains in section 2.1. In this section, we will further present a formulation which incorporates sample-wise computations. Even though we consider deterministic problems here in this chapter, we will describe a multi-sample formulation of the triangulation to later use it for our parallelization approach. In the second section 2.2 of this chapter, we will construct the finite element spaces, where we will search for the numerical solution approximating the true solution of the imposed problem. Depending on the model, some spaces are applicable or more useful than others. Thus, in the third section 2.3, we will discuss the application of the introduced finite element spaces to the imposed model problems and introduce their weak formulations and the assembling of the discretized equations. By searching for the discrete solution, we will admit an approximation error to the true solution, which was in the introduction denoted by $\mathrm{err}_{\mathrm{disc}}$. The central question in the numerical analysis is how the error behaves with respect to the cell diameter or the time discretization step size. Therefore, we present results answering this question in section 2.3, too. To solve the assembled and distributed systems of equations, we also introduce the employed numerical solvers briefly in 2.4, before we give a deeper insight in our software M++, its architecture and its core implementation ideas in 2.5. Additionally, we want to note that git-references to the implementation (as this one to the main page of the repository {2}) are continuously given throughout the chapter to give a technical perspective of the presented theory.

In the following, we will denote with $u \in V$ the true solution to the PDE and use $\ell \in \mathbb{N}_0$ as subscript for the numerical solution $u_\ell \in V_\ell$, where V_ℓ is a finite dimensional function space. Instead of the commonly used h as index for the numerical solution, we chose this notation as later on in chapter 3, we will introduce and use multi-level methods. To keep the notation lean and still be able to depict the mesh hierarchies over several *levels*, we settled on ℓ and only use h_ℓ, if we explicitly mean the cell diameter.

2.1 Multi-mesh Triangulation

Finite element methods (FEMs) compute an approximation u_ℓ to some true solution $u \in V$ of a PDE in a finite dimensional function space V_ℓ. As these PDEs are defined on a physical domain $D \subset \mathbb{R}^{\mathbf{d}}$ and potentially on a time interval $(0, T]$, we have to discretize these physical spaces in order to define the finite dimensional function spaces V_ℓ. Therefore, the domain D is decomposed in finitely many subdomains K (e.g. triangles) which are collected in a set $\mathcal{K}$, such that

$$\overline{D} = \bigcup_{K \in \mathcal{K}} \overline{K} \quad \text{and} \quad \overset{\circ}{K} \cap \overset{\circ}{K'} = \emptyset \quad \text{for} \quad K \neq K'. \tag{2.1}$$

We will further specify this triangulation in the following paragraph. However, as it turns out in practical applications, the resulting data structure and the algorithms acting on it require a huge amount of computational resources as the domain gets decomposed in finer and finer subdomains. The necessary parallelization over a set of processes $\mathcal{P}$, for which we assume that its cardinality is a power of two, and the corresponding distribution to cope with the computational load is also based on a domain decomposition. Simply speaking, we will assign to each process $P \in \mathcal{P}$ a fraction of the domain D denoted by D_P, such that

$$\overline{D} = \bigcup_{P \in \mathcal{P}} \overline{D}_P = \bigcup_{P \in \mathcal{P}} \bigcup_{K \in \mathcal{K}_P} \overline{K},$$

where $\mathcal{K}_P$ is the collection of subdomains on a single process P. In the second paragraph of this section, we will follow the notation of [28] and formally explain the parallelization process as well as its implementation. Moreover, as we want to consider randomized and parametric PDEs later on, we will also present a parallelization over a set of data realizations $\{\mathbf{y}^{(m)}\}_{m=1}^M$ on distributed memory. We refer for details to section 3.2 and consider $\{\mathbf{y}^{(m)}\}_{m=1}^M$ simply as a collection of data points. This is motivated by the non-intrusiveness of the UQ methods. Likewise to shrinking subdomains K, the computational cost increases with the amount of demanded realizations M. This requires to employ an even more refined approach to the parallelization, namely to distribute the set of processes on $\{\mathbf{y}^{(m)}\}_{m=1}^{M} \times \overline{D}$. This is done by choosing for each $m = 1, \dots, M$ a subset of processes $\mathcal{P}_k^{(m)} \in \mathcal{P}$ and distribute the domain on this subset. All $\mathcal{P}_k^{(m)}$ are disjoint and of size $|\mathcal{P}_k^{(m)}| = 2^k$ where $k \in \mathbb{N}_0$ is chosen such that the inter process communication is minimized. We will motivate this in detail in the last paragraph of this section. The final parallelism is highly integrated and closely implemented to the hardware, however, the true benefit of this method can only be explained as we have discussed multi-level estimators in section 3.5. Other approaches to solve a similar problems are given in [16, 79, 237, 240].

2.1.1 Triangulation and Reference Cells

We assume that D is either a polygonal domain in $\mathbb{R}^{\mathbf{d}}$ or space-time domain in $\mathbb{R}^{\mathbf{d}+1}$ such as in the introduction. A triangulation is a spatial (space-time) decomposition of this domain D into a finite set of polyhedra $\mathcal{K}$ according to (2.1) which further defines a set of vertices $\mathcal{V}$, a set of faces $\mathcal{F}$ and a set of edges $\mathcal{E}$. A single polyhedron $K \in \mathcal{K}$ is commonly called a cell, defined by the convex hull of the subset $\mathcal{V}_K \subset \mathcal{V}$, where $\mathcal{V}_K$ ist the set of vertices of a single cell. The boundary ∂K of each cell consists of finitely many faces $\mathcal{F}_K = \{F \in \mathcal{F} \colon \mathcal{V}_F \subset \mathcal{V}_K\}$ being $\max\{\mathbf{d}-1, 0\}$ dimensional hyper planes. Likewise, the boundary of each face ∂F consist of finitely many edges $\mathcal{E}_F = \{E \in \mathcal{E} \colon \mathcal{V}_E \subset \mathcal{V}_F \subset \mathcal{V}_K\}$, where $\mathcal{V}_F$ and $\mathcal{E}_E$ aber are the sets of vertices for face F and edge E, respectively. This defines a mesh as a set of vertices, cells, faces and edges as well as a cell as a set of the corresponding subsets, thus

$$\mathcal{M} = \{\mathcal{V}, \mathcal{K}, \mathcal{F}, \mathcal{E}\} \quad \text{and} \quad \mathcal{K} \ni K = \{\mathcal{V}_K, \mathcal{F}_K, \mathcal{E}_K\}.$$

Our implementation of a mesh as set of sets is given in {3}. The described triangulation is admissible, if for every $K, K' \in \mathcal{K}$ the convex hull of the intersection equals the intersection of the convex hulls of the cells, i.e.,

$$\operatorname{conv}(\mathcal{V}_K \cap \mathcal{V}_{K'}) = \operatorname{conv}(\mathcal{V}_K) \cap \operatorname{conv}(\mathcal{V}_{K'}). \tag{2.2}$$

Therefore, we consider the triangulation to be admissible if on both sides of equation (2.2) the intersection is either empty, a vertex, a face or an edge. We will depict the unit normal vector on the face $F = \overline{K} \cap \overline{K'}$ pointing outwards of K by $\mathbf{n}_K$. Additionally, we define for each cell the diameter $h_K := \sup\{|\mathbf{x} - \mathbf{y}| : \mathbf{x}, \mathbf{y} \in K\}$ and with that the maximum $h_{\max} := \max\{h_K \colon K \in \mathcal{K}\}$ and minimum $h_{\min} := \min\{h_K \colon K \in \mathcal{K}\}$ mesh widths. An example triangulation of a two-dimensional domain D with triangular cells and different cell diameters h_K is depicted in figure 2.1.

Furthermore, every cell $K \in \mathcal{K}$ is associated with a reference cell $\hat{K}$ and then purely determined by a bijective and orientation preserving mapping

$$\varphi_K \colon \operatorname{conv} \mathcal{V}_{\hat{K}} \to \operatorname{conv} \mathcal{V}_K \quad \text{with} \quad \varphi_K(\hat{\mathbf{z}}_i) = \mathbf{z}_{K,i} \quad i \in \{1, \dots |\mathcal{V}_{\hat{K}}|\}.$$

With that we can define a regular triangulation, namely if all φ_K are regular transformations, i.e.,

$$\exists c > 0 \colon \left|\varphi_K^{-1}\right| \leq c h_K^{-1} \quad \forall K \in \mathcal{K}. \tag{2.3}$$

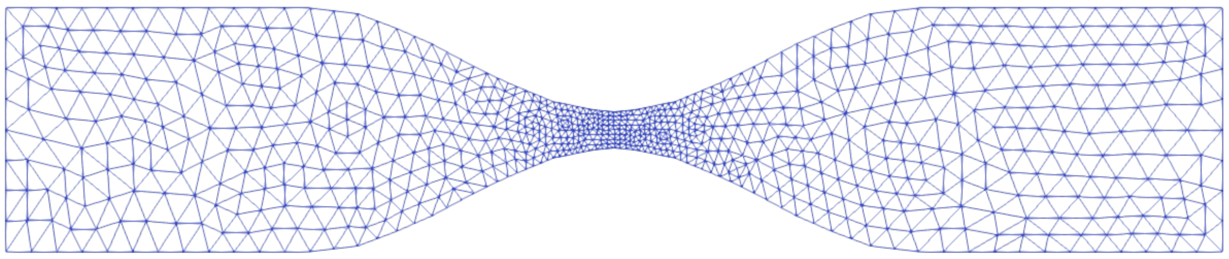


FIGURE 2.1: Admissiblae triangulation of a narrowing domain with different cell diameters h_K.

If the cells are very pointy this results in poor bounds with respect to the mesh width by (2.3) and thereby, in high computational costs. However, as we mostly use uniformly refined grids, i.e. $h_{\max} = h_{\min}$, we omit a detailed explanation. For a simplified notation, we restrict ourselves to linear affine functions $\varphi_K = \mathbf{z}_{K,1} + J_K\xi$ with $\mathbf{z}_{K,1}$ as a corner of K and $J_K = \mathrm{D}\varphi_K$ with $|\det(J_K)| = |K| / |\hat{K}|$ (confer [81, chapter 1] for the non-affine case and further properties). A quick overview on the different cell types in use within this thesis and M++ is given in table 2.1.

Furthermore, to be able to quickly access with $\mathcal{O}(1)$ all cells $\mathcal{K}$, faces $\mathcal{F}$ and edges $\mathcal{E}$, we assign to each object a hash of their midpoints $\mathbf{z}_K, \mathbf{z}_F, \mathbf{z}_E$

$$\widetilde{K} = \texttt{Hash}\underbrace{\left(\frac{1}{|\mathcal{V}_K|}\sum_{\mathbf{z}\in\mathcal{V}_K}\mathbf{z}\right)}_{=\mathbf{z}_K}, \quad \widetilde{F} = \texttt{Hash}\underbrace{\left(\frac{1}{|\mathcal{V}_F|}\sum_{\mathbf{z}\in\mathcal{V}_F}\mathbf{z}\right)}_{=\mathbf{z}_F}, \quad \widetilde{E} = \texttt{Hash}\underbrace{\left(\frac{1}{|\mathcal{V}_E|}\sum_{\mathbf{z}\in\mathcal{V}_E}\mathbf{z}\right)}_{=\mathbf{z}_E}$$

and store the sets $\mathcal{K}$, $\mathcal{F}$ and $\mathcal{E}$ as hashmaps {4}. A detailed description of this concept of distributed point objects and the exploitation of hashmaps can be found for instance in [257] and [256].

Lastly we remark, by checking the Euler relation [81], e.g. for a two-dimensional mesh without holes, allows us to verify an implementation of a triangulation with

$$|\mathcal{K}| - |\mathcal{E}| + |\mathcal{V}| = 1, \quad |\mathcal{V} \cap \partial D| - |\mathcal{E} \cap \partial D| = 0 \quad \text{and} \quad 2\,|\mathcal{E}| - |\mathcal{E} \cap \partial D| = |\mathcal{V}_K| \cdot |\mathcal{K}|\,.$$

Again, we refer to the test-code {5} and to [81] for the Euler relations in three dimensions.

Cell type	Dimension	Corners $\lvert\mathcal{V}_K\rvert$	Faces $\lvert\mathcal{F}_K\rvert$	Edges $\lvert\mathcal{E}_K\rvert$	Implementation
Interval	1	2	2	2	{6}
Triangle	2	3	3	3	{7}
Quadrilateral	2	4	4	4	{8}
Hexahedron	3	8	6	12	{9}
ST-interval	$1+1$	$2 \cdot 2$	$2+2$	$2 \cdot 1 + 2$	{10}
ST-quadrilateral	$2+1$	$2 \cdot 4$	$4+2$	$2 \cdot 1 + 2$	{11}

TABLE 2.1: Overview on different cell-types.

2.1.2 Single-mesh Parallelization and Refinement

In order to deal with high memory and computational demands, the domain D is decomposed into disjoint subdomains D_P (which might possess some overlap for the discontinuous Galerkin (dG) discretization) and the mesh $\mathcal{M}$ is distributed on a set of processes $\mathcal{P} = \{0, \dots, |\mathcal{P}| - 1\}$. The distribution is determined by $\pi \colon \mathcal{Z} \to 2^{\mathcal{P}}$, which maps a collection of points $\mathcal{Z}$ to the power set of processes (confer definition A.7 of a power set in the appendix). For example, if the domain is decomposed on two processors P_1 and P_2 the power set is $\{\emptyset, \{P_1\}, \{P_2\}, \{P_1, P_2\}\}$. For most of the points $z \in \mathcal{Z}$ the value of the mapping is either $\{P_1\}$ or $\{P_2\}$. However, for points on the processor boundary the value is $\{P_1, P_2\}$. This mapping can for example be constructed by recursive coordinate bisection (RCB), dividing the domain recursively into two parts along a certain axis. The collection $\mathcal{Z}$ consists of all nodal points which will be defined in section 2.2, however for now it is just the collection of all vertices and midpoints to faces, edges and cells. The resulting *local* mesh $\mathcal{M}_P = \{\mathcal{V}_P, \mathcal{K}_P, \mathcal{F}_P, \mathcal{E}_P\}$ is the collection of

$$\begin{aligned} \mathcal{V}_P &= \{\mathbf{z} \in \mathcal{V} \colon P \in \pi(\mathbf{z})\}, & \mathcal{F}_P &= \{F \in \mathcal{F} \colon P \in \pi(\mathbf{z}_F)\}, \\ \mathcal{K}_P &= \{K \in \mathcal{K} \colon P \in \pi(\mathbf{z}_K)\}, & \mathcal{E}_P &= \{E \in \mathcal{E} \colon P \in \pi(\mathbf{z}_E)\}, \end{aligned}$$

whereas $\mathbf{z}_K, \mathbf{z}_F, \mathbf{z}_E$ are the midpoints of the cells, faces and edges. The parallelized mesh is then given by $\mathcal{M}_{\mathcal{P}} = \bigcup_{P \in \mathcal{P}} \mathcal{M}_P$.

Furthermore, in order to be able to use multi-level algorithms later on in section 3.5, a mesh hierarchy $\mathcal{M}_{\ell=0,\mathcal{P}} \subset \mathcal{M}_{\ell=1,\mathcal{P}} \subset \cdots \subset \mathcal{M}_{\ell=L,\mathcal{P}}$ has to be constructed. This process is guided by cell dependent refinement rules and happens on the local meshes $\mathcal{M}_{\ell,P}$ only.

2.1.3 Multi-mesh Parallelization

Lastly, we want to introduce our multi-mesh parallelization. As it will turn out in chapter 3, solving the initially proposed problems will require the evaluation of many realizations, potentially even on different levels. This is computationally a highly challenging task and requires efficient load balancing. Classical parallelization over the domain as described in the previous paragraph or solving for the individual realizations with the input data $\mathbf{y}^{(m)}$ in parallel will not be sufficient in order to cope with the heterogeneous computational demand of different samples, especially in a multi-level framework.

In order to avoid further details about the UQ algorithms, we only want to regard the following problem: Approximate M_ℓ times a PDE on level ℓ with a fixed amount of processes $|\mathcal{P}|$. As a start, we investigate two edge cases of this problem. Firstly, we only want to approximate a single sample. Then clearly, the best parallelization is given by the previously described domain decomposition resulting in a single, parallelized mesh $\mathcal{M}_{\mathcal{P}}$ over the whole domain D. An illustration of this case is given in figure 2.2 on the very left for $|\mathcal{P}| = 4$. Secondly, if the sample amount equals the amount of processes, a minimal communication, and thus an optimal parallelization, is achieved by assigning each process its very own unparallelized mesh, resulting in a set of meshes (confer figure 2.2 on the very right)

$$M_\ell = |\mathcal{P}| \quad \Rightarrow \quad \mathcal{M}_{\mathcal{P}} = \left\{\mathcal{M}_P^{(m)}\right\}_{m=1}^{M_\ell}, \quad m = P \in \mathcal{P}.$$

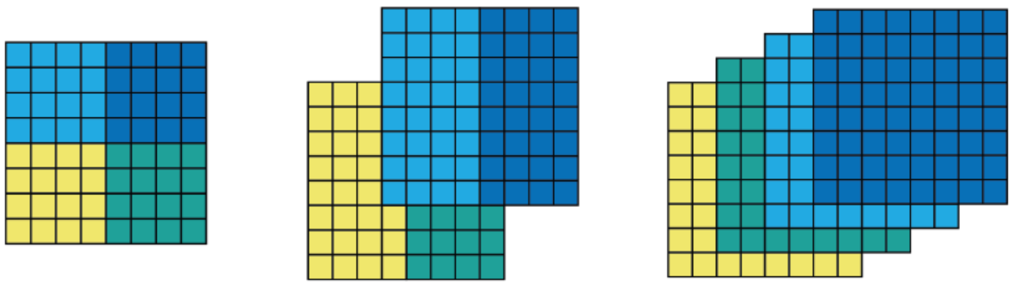


FIGURE 2.2: Parallelization approaches for $D = (0,1)^2$ and $|\mathcal{P}| = 4$ according to (2.4). First, only on spatial domain D for $M_\ell = 1$ with $k = 2$. Second, mixed parallelization for $M_\ell = 2$ with $k = 1$. Last, only for $M_\ell = 4$ with $k = 0$.

Moving on to the general case, where we want to compute more samples than we have processes at hand $M_\ell \geq |\mathcal{P}|$. Then, we simply split the samples in batches of size less or equal to M_ℓ (rounding up to $|\mathcal{P}|$, if possible in UQ algorithm) and then sequentially compute each batch in parallel on the set of multiple meshes $\{\mathcal{M}_P\}_{P=1}^{|\mathcal{P}|}$.

Lastly, we have to consider the case, as depicted in the middle of figure 2.2, where $1 < M_\ell < |\mathcal{P}|$. Here, we search for $k \in \mathbb{N}_0$, satisfying

$$2^k \leq \frac{|\mathcal{P}|}{M_\ell} < 2^{k+1} \quad \Rightarrow \quad \mathcal{M}_{\mathcal{P}} = \left\{ \mathcal{M}_{\mathcal{P}_k}^{(m)} \right\}_{m=1}^{M_\ell} \quad \text{with} \quad \mathcal{P} = \bigsqcup_{m=1}^{M_\ell} \mathcal{P}_k^{(m)} \quad \text{and} \quad |\mathcal{P}_k| = 2^k \tag{2.4}$$

determining the distribution of the processes over samples $\mathbf{y}^{(m)}$ as well as over the spatial domain D. In particular, the processes are grouped as in the following scheme:

$$\begin{aligned}
2^{\mathcal{P}} \supset \{ \quad & \{0\},\{1\},\{2\},\{3\},\{4\},\{5\},\{6\},\{7\},\dots,\{|\mathcal{P}|-1\}, && \Leftarrow \quad k = 0 \\
& \{0,1\},\{2,3\},\{4,5\},\{6,7\},\dots,\{|\mathcal{P}|-2,|\mathcal{P}|-1\}, && \Leftarrow \quad k = 1 \\
& \{0,1,2,3\},\{4,5,6,7\},\dots,\{|\mathcal{P}|-4,\dots,|\mathcal{P}|-1\}, && \Leftarrow \quad k = 2 \\
& \left\{ \tfrac{i|\mathcal{P}|}{2^k},\dots,\tfrac{(i+1)|\mathcal{P}|}{2^k}-1 \right\}_{i=0,\dots,2^k-1}, && \quad\quad \vdots \\
& \left\{0,\dots,\tfrac{|\mathcal{P}|}{4}-1\right\},\dots,\left\{\tfrac{3|\mathcal{P}|}{4},\dots,|\mathcal{P}|-1\right\}, && \Leftarrow \quad k = \log_2|\mathcal{P}|-2 \\
& \left\{0,\dots,\tfrac{|\mathcal{P}|}{2}-1\right\},\left\{\tfrac{|\mathcal{P}|}{2},\dots,|\mathcal{P}|-1\right\}, && \Leftarrow \quad k = \log_2|\mathcal{P}|-1 \\
& \{0,1,2,3,4,5,6,7,\dots,|\mathcal{P}|-1\} \quad \}. && \Leftarrow \quad k = \log_2|\mathcal{P}|
\end{aligned}$$

In the actual implementation {12} the index k in the scheme counts from the bottom up instead from the top down. The reason for this is that we split the MPI communicators successively starting from the *world communicator*. The world communicator contains all processes and is usually associated with the index zero. We refer to the sections 2.5 and 3.6 for a short explanation on MPI.

As a result, we can expand the definition of the distribution mapping π by the sample id m

$$\pi \colon \{1,\dots,M_\ell\} \times \mathcal{Z} \to 2^{\mathcal{P}} \tag{2.5}$$

and thereby assign to each sample and each geometric object a set of processes. In the next section, we will describe how we define the finite element space on top of this distributed data structure of geometric objects. Further properties of this parallelization approach will be described in chapter 3, as we will explore the different UQ algorithms. As a central piece of the method development presented in this thesis, we will also conduct several numerical experiments proving its applicability, its adaptivity and its scalability on HPC clusters in chapter 4.

2.2 Finite Element Spaces

We briefly present the construction of the used finite element spaces within this thesis. The notation follows [28], however the theory is mostly restated from [44] and [81]. At the end of the section, we will exploit the multi-mesh parallelization and define finite element spaces spanning over the domain of the input data.

The deterministic model problems 1.1-1.3 can be expressed in a weak formulation (we refer to section 2.3 for details) with solutions lying in infinite dimensional function spaces V. To find an approximation, we need a way to construct finite dimensional subspaces V_ℓ. Following Ciarlet [57], we restate the definition of a *finite element*.

Definition 2.1. A *finite element* is a triplet (K, V_K, V_K'), which is characterized by

a) a *cell* $K \in \mathcal{K}$,

b) a vector space of local *shape functions* V_K defined on the cell K,

c) a set of *nodal variables* or *local degrees of freedom* (dofs) being a dual basis to V_K.

In order to define the basis for V_K, we consider to the reference cell $\hat{K}$ and introduce a set of reference *nodal points* $\hat{\mathbf{z}} \in \hat{\mathcal{Z}}$, which can for example be vertices, cell, face or edge midpoints or some other points placed in the reference cell $\hat{K}$. Then, we associate each nodal point with an ansatz function $\hat{\psi}_{\hat{\mathbf{z}}}$ and a nodal variable $\hat{\psi}'_{\hat{\mathbf{z}}}$. With that we can construct a finite element on an arbitrary cell $K \in \mathcal{K}$ by

$$V_K = \operatorname{span}\left\{\psi_{K,\mathbf{z}} \colon \psi_{K,\mathbf{z}} = \hat{\psi}_{\hat{\mathbf{z}}} \circ \varphi_K^{-1},\, \hat{\psi}_{\hat{\mathbf{z}}} \in \hat{V}_{\hat{K}}\right\} \quad \text{with} \quad V_K \ni v = \sum_{\mathbf{z} \in \mathcal{Z}_K} \left\langle \psi'_{K,\mathbf{z}}, v \right\rangle \psi_{K,\mathbf{z}},$$

where the nodal points $\mathbf{z} \in \mathcal{Z}_K$ are given by the transformed reference nodal points $\hat{\mathbf{z}} \in \hat{\mathcal{Z}}$. The ansatz functions and nodal variables are chosen such that they satisfy a Kronecker property

$$\left\langle \hat{\psi}'_{\hat{\mathbf{z}}_i}, \hat{\psi}_{\hat{\mathbf{z}}_j} \right\rangle = \delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \quad, \quad i, j = 1, \dots, |\hat{\mathcal{Z}}| \quad \Rightarrow \quad \left\langle \psi'_{\mathbf{z}_i}, \psi_{\mathbf{z}_j} \right\rangle = \delta_{ij}. \tag{2.6}$$

The coefficients in the linear combination (the dofs) are determined by an *interpolation operator*. Examples are given by:

$$\begin{aligned}
\text{Point evaluations: } \langle \psi'_{\mathbf{z}}, v\rangle &= v(\mathbf{z}), \quad \mathbf{z} \in \mathcal{Z}_K \\
\text{Element integrals: } \langle \psi'_{\mathbf{z}}, v\rangle &= \int_K v(\mathbf{x})\,\mathrm{d}\mathbf{x}, \quad K \in \mathcal{K} \\
\text{Face averages of normal component: } \langle \psi'_{\mathbf{z}}, v\rangle &= \int_F v(\mathbf{x}) \cdot \mathbf{n}_F \,\mathrm{d}\mathbf{a}, \quad F \in \mathcal{F}_K
\end{aligned}$$

By collecting all local nodal points $\mathcal{Z}_K$ in a global set $\mathcal{Z}$, we can move from the local space of shape functions V_K to the global finite element space

$$V_\ell = \operatorname{span}\{\psi_{\ell,\mathbf{z}} \colon \psi_{\ell,\mathbf{z}}|_K = \psi_{K,\mathbf{z}} \in V_K \text{ for all } K \in \mathcal{K}_\ell \text{ and all } \mathbf{z} \in \mathcal{Z}\}.$$

Concluded, by defining the geometric shape of the reference cell $\hat{K}$, selecting the nodal points $\hat{\mathbf{z}} \in \hat{\mathcal{Z}}$, the shape functions $\hat{\psi}_{\hat{\mathbf{z}}}$ and the nodal variables $\hat{\psi}'_{\hat{\mathbf{z}}}$, we can define a large class of FEM for which we will depict some instances in further detail.

In the following, $\mathbb{P}_{\mathbf{p}}(K, \mathbb{R})$ is the polynomial space of degree at most $\mathbf{p} \in \mathbb{N}_0$ on the local cell $K \subset \mathbb{R}^{\mathbf{d}}$ taking values in $\mathbb{R}$, and $\mathbb{Q}_{\mathbf{p}}(K, \mathbb{R})$ depicts the tensor product space of local Lagrange polynomials. Hence $\mathbb{P}_{\mathbf{p}}(K, \mathbb{R})$ and $\mathbb{Q}_{\mathbf{p}}(K, \mathbb{R})$ are given by

$$\begin{aligned}
\mathbb{P}_{\mathbf{p}}(K, \mathbb{R}) &= \left\{ \sum_{\substack{0 \le p_1, \dots, p_d \le \mathbf{p} \\ p_1 + \dots + p_d \le \mathbf{p}}} c_{p_1 \dots p_d} \cdot x_1^{p_1} \cdot \ldots \cdot x_d^{p_d} \colon c_{p_1 \dots p_d} \in \mathbb{R} \right\} \\
\mathbb{Q}_{\mathbf{p}}(K, \mathbb{R}) &= \left\{ \sum_{0 \le p_1, \dots, p_d \le \mathbf{p}} c_{p_1 \dots p_d} \cdot x_1^{p_1} \cdot \ldots \cdot x_d^{p_d} \colon c_{p_1 \dots p_d} \in \mathbb{R} \right\}
\end{aligned}$$

with

$$\dim \mathbb{P}_{\mathbf{p}} = \begin{cases} \mathbf{p} + 1 & \text{on intervals} \\ \frac{1}{2}(\mathbf{p}+1)(\mathbf{p}+2) & \text{on triangles} \\ \frac{1}{6}(\mathbf{p}+1)(\mathbf{p}+2)(\mathbf{p}+3) & \text{on tetrahedrons} \end{cases} \qquad \dim \mathbb{Q}_{\mathbf{p}} = \begin{cases} (\mathbf{p}+1)^2 & \text{on quadrilaterals} \\ (\mathbf{p}+1)^3 & \text{on hexahedrons} \\ (\mathbf{p}+1)^3 & \text{on ST-quadrilaterals} \end{cases}$$

For many discretizations the following definition will be useful.

Definition 2.2. For $F = K \cap K'$, we define for vector valued functions $\mathbf{v}_\ell$ and scalar valued functions v_ℓ

$$[\![\mathbf{v}_\ell]\!]_F = \mathbf{v}_\ell|_{K'} - \mathbf{v}_\ell|_K \in \mathbb{R}^{\mathbf{d}}, \quad [\![v_\ell]\!]_F = (v_\ell|_{K'} - v_\ell|_K)\mathbf{n}_{K_F} \in \mathbb{R}^{\mathbf{d}} \quad \{\!\{v_\ell\}\!\}_F = \tfrac{1}{2}(v_\ell|_K + v_\ell|_{K'}) \in \mathbb{R} \tag{2.7}$$

the *jump* and the *face average* at the face $F \in \mathcal{F}$.

2.2.1 Lagrange Finite Elements

The standard finite element space is given through continuous ($[\![v_\ell]\!]_F = 0$ for all $F \in \mathcal{F} \setminus \partial D$), local Lagrange polynomials of degree $\mathbf{p} \geq 1$

$$V^{\mathrm{Lag}}_{\ell,\mathbf{p}\geq 1} = \begin{cases} \left\{ v_\ell \in \mathrm{C}^0(\overline{D})\colon v_\ell|_K \in \mathbb{P}_{\mathbf{p}\geq 1},\, \forall K \in \mathcal{K} \right\}, & \text{for intervals, triangles and tetrahedrons} \\ \left\{ v_\ell \in \mathrm{C}^0(\overline{D})\colon v_\ell|_K \in \mathbb{Q}_{\mathbf{p}\geq 1},\, \forall K \in \mathcal{K} \right\}, & \text{for quadrilaterals, hexahedrons and ST-cells} \end{cases}$$

where the polynomial basis to this space is such that the Kronecker property is satisfied and where the (local) degrees of freedom are point evaluations with the (local) interpolation operator

$$\Pi^{\mathrm{Lag}}_{K,\mathbf{p}\geq 1}\colon C^0(\overline{K}) \ni v \longmapsto \sum_{\mathbf{z}\in\mathcal{Z}_K} v(\mathbf{z})\psi_{K,\mathbf{z}} \in V^{\mathrm{Lag}}_{K,\mathbf{p}\geq 1} \quad \Rightarrow$$
$$\Pi^{\mathrm{Lag}}_{\ell,\mathbf{p}\geq 1}\colon C^0(\overline{D}) \ni v \longmapsto \sum_{\mathbf{z}\in\mathcal{Z}} v(\mathbf{z})\psi_{\ell,\mathbf{z}} \in V^{\mathrm{Lag}}_{\ell,\mathbf{p}\geq 1}.$$

The interpolation $\Pi^{\mathrm{Lag}}_{\ell,\mathbf{p}\geq 1} v$ is unique and interpolates $v \in V$ in the nodal points, whereas functions $v_\ell \in V^{\mathrm{Lag}}_{\ell,\mathbf{p}\geq 1}$ are invariant under this interpolation $\Pi^{\mathrm{Lag}}_{\ell,\mathbf{p}} v_\ell = v_\ell$. Since $v_\ell \in V^{\mathrm{Lag}}_{\ell,\mathbf{p}\geq 1}$ is in $\mathrm{L}^2(D)$ and continuous, this implies the following lemma (cf. for further details [81, Propostion 1.74]).

Lemma 2.3. *For the FE-space with Lagrangian ansatz functions we have* $V^{\mathit{Lag}}_{\ell,\mathbf{p}\geq 1} \subset \mathrm{H}^1(D)$.

Example 2.4. a) On a one-dimensional domain, the basis for $V^{\mathrm{Lag}}_{\ell,\mathbf{p}=1}$ is spanned by *hat functions* which are associated with each nodal point in the domain.

b) On the reference triangle, we span the space of linear functions with $\hat{\psi}_{\hat{\mathbf{z}}_1}(\mathbf{x}) = 1 - x_1 - x_2$, $\hat{\psi}_{\hat{\mathbf{z}}_2}(\mathbf{x}) = x_1$ and $\hat{\psi}_{\hat{\mathbf{z}}_3}(\mathbf{x}) = x_2$.

c) On the reference tetrahedron, we span the space of linear functions with $\hat{\psi}_{\hat{\mathbf{z}}_1}(\mathbf{x}) = 1 - x_1 - x_2 - x_3$, $\hat{\psi}_{\hat{\mathbf{z}}_2}(\mathbf{x}) = x_1$, $\hat{\psi}_{\hat{\mathbf{z}}_3}(\mathbf{x}) = x_2$ and $\hat{\psi}_{\hat{\mathbf{z}}_4}(\mathbf{x}) = x_3$.

d) On the reference quadrilateral, we span the space of bilinear functions with $\hat{\psi}_{\hat{\mathbf{z}}_1}(\mathbf{x}) = (1-x_1)(1-x_2)$, $\hat{\psi}_{\hat{\mathbf{z}}_2}(\mathbf{x}) = x_1(1-x_2)$, $\hat{\psi}_{\hat{\mathbf{z}}_3}(\mathbf{x}) = x_1 x_2$ and $\hat{\psi}_{\hat{\mathbf{z}}_4} = (1-x_1)x_2$.

e) On the reference hexahedron, we span the space of tri-polynomial functions with a tensor product of Lagrange polynomials in one dimension

$$\left\{ \hat{\psi}_{\hat{z}_i} \right\}_{i=0}^{\mathbf{p}} \quad \text{with} \quad \hat{\psi}_{\hat{z}_i}(\hat{x}) = \prod_{\substack{0\leq p\leq \mathbf{p} \\ p\neq i}} \frac{\hat{x} - \hat{z}_p}{\hat{z}_i - \hat{z}_p} \quad \Rightarrow \quad \hat{\psi}_{\hat{\mathbf{z}}_j}(\hat{\mathbf{x}}) = \bigotimes_{d=1}^{3} \hat{\psi}_{\hat{z}_j}(\hat{x}_d).$$

We refer to figure 2.3 for a visualization of the nodal points. It can be easily checked that all examples above satisfy the Kronecker property. At this point, we want to emphasize that everything above also holds for vector valued functions by replacing $\mathbb{P}_{\mathbf{p}}(K)$ with $\mathbb{P}_{\mathbf{p}}(K, \mathbb{R}^J)$ and $\mathbb{Q}_{\mathbf{p}}(K)$ with $\mathbb{Q}_{\mathbf{p}}(K, \mathbb{R}^J)$, respectively. Investigating the properties of the interpolation operator on Sobolev spaces is a crucial part in the error analysis in FEM. Thereby, we state the following theorem.

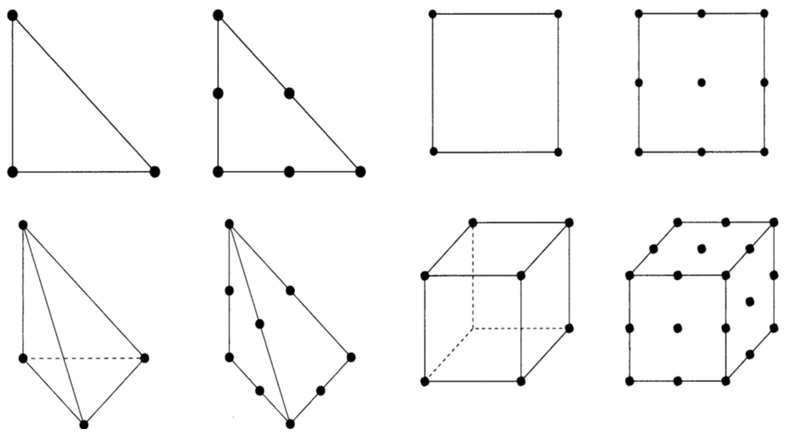


FIGURE 2.3: Triangular, quadrilateral, tetrahedral and hexahedral reference cells with nodal points for Lagrange ansatz function of degree $\mathbf{p} = 1$ and degree $\mathbf{p} = 2$. Figures rearranged from [81].

Theorem 2.5. *Consider the interpolation operator* $\Pi^{\mathrm{Lag}}_{\ell,\mathbf{p}\geq 1} \colon \mathrm{H}^{s+1}(D) \to V^{\mathrm{Lag}}_{\ell,\mathbf{p}\geq 1}$ *and let* $0 \leq k \leq \min\{s,\mathbf{p}\}+1$. *The interpolation estimate is given by*

$$\left\| v - \Pi^{\mathrm{Lag}}_{\ell,\mathbf{p}\geq 1} v \right\|_{\mathrm{H}^k(D)} \lesssim h^{\min\{s,\mathbf{p}\}+1-k} \left| v \right|_{\mathrm{H}^{\min\{s,\mathbf{p}\}+1}(D)} .$$

For sufficiently smooth functions, i.e., $s \geq \mathbf{p}$, *the estimate can be simplified for example in the* H^1*- and the* L^2*-norm by*

$$\left\| v - \Pi^{\mathrm{Lag}}_{\ell,\mathbf{p}\geq 1} v \right\|_{\mathrm{L}^2(D)} \lesssim h^{\mathbf{p}+1} \left| v \right|_{\mathrm{H}^{\mathbf{p}+1}(D)} \quad \textit{and} \quad \left\| v - \Pi^{\mathrm{Lag}}_{\ell,\mathbf{p}\geq 1} v \right\|_{\mathrm{H}^1(D)} \lesssim h^{\mathbf{p}} \left| v \right|_{\mathrm{H}^{\mathbf{p}+1}(D)} .$$

Proof. This result can be found in any book about FEM. For example, in [81, section 1.5] this interpolation result is first developed locally (theorem 1.103) and then extended globally on an affine mesh in corollary 1.109 and 1.110. □

The interpolation result can be expanded with the trace theorem [42, Chapter 3] to obtain interpolation estimates in the $\mathrm{L}^2(\partial D)$-norm.

Theorem 2.6. *Consider the interpolation operator* $\Pi^{\mathrm{Lag}}_{\ell,\mathbf{p}\geq 1} \colon \mathrm{H}^{\mathbf{p}+1}(D) \to V^{\mathrm{Lag}}_{\ell,\mathbf{p}\geq 1}$ *and let* $u \in \mathrm{H}^{\mathbf{p}+1}(D)$. *The interpolation estimate on the boundary in the* $\mathrm{L}^2(\partial D)$*-norm is given by*

$$\left\| v - \Pi^{\mathrm{Lag}}_{\ell,\mathbf{p}\geq 1} v \right\|_{\mathrm{L}^2(\partial D)} \lesssim h^{\mathbf{p}+1/2} \left| v \right|_{\mathrm{H}^{\mathbf{p}+1}(D)} .$$

2.2.2 Raviart-Thomas Elements

Raviart-Thomas (RT) elements are often in use to approximate functions which represent velocities and thus, they often find applications in fluid mechanics. Thereby, a natural choice for the degrees of freedom are the averages of the normal component across the faces of a cell. This defines the interpolation operator as

$$\Pi_K^{\mathrm{RT}} \colon V(K) \ni \mathbf{v} \longmapsto \sum_{F\in\mathcal{F}_K} \left(\int_F \mathbf{v}\cdot\mathbf{n}_F \mathrm{d}a\right) \boldsymbol{\psi}_{K,\mathbf{z}} \in V_K^{\mathrm{RT}} \quad \Rightarrow$$
$$\Pi_\ell^{\mathrm{RT}} \colon V \ni \mathbf{v} \longmapsto \sum_{F\in\mathcal{F}} \left(\int_F \mathbf{v}\cdot\mathbf{n}_F \mathrm{d}a\right) \boldsymbol{\psi}_{\ell,\mathbf{z}} \in V_\ell^{\mathrm{RT}}.$$

The domain of the (local) interpolation operator is taken as $\mathrm{H}(\mathrm{div}, K)$ and $\mathrm{H}(\mathrm{div}, D)$ (confer definition A.3 c)), respectively, which assures that the face integral is meaningful and that the face normal components are continuous ($[\![\mathbf{v}_\ell]\!]_F = \mathbf{0}$ for all $F \in \mathcal{F} \setminus \partial D$). Here, we will only consider the case $\mathbf{d} = 2$, three-dimensional elements can for example be found in [46]. Furthermore, we only consider ansatz functions of the form

$$\boldsymbol{\psi}_{F_i}(\mathbf{x}) = \frac{1}{\mathbf{d}\,|K|}(\mathbf{x} - \mathbf{z}_i), \quad 0 \le i \le \mathbf{d},$$

i.e., of order one, which can also be generalized to a higher order. With these ansatz functions, we construct the local finite element Raviart-Thomas space with

$$V^{\mathrm{RT}}(K) = \left\{\mathbf{v}_K \in \mathbb{P}_1(K, \mathbb{R}^{\mathbf{d}}) \colon \mathbf{v}_K(\mathbf{x}) = \mathbf{a} + b\mathbf{x} \colon (\mathbf{a}, b) \in \mathbb{R}^{\mathbf{d}} \times \mathbb{R}\right\}$$

and the global finite element space then by $V_\ell^{\mathrm{RT}} = \left\{\mathbf{v}_\ell \in \mathrm{H}(\mathrm{div}, D) : \mathbf{v}_K \in V^{\mathrm{RT}}(K)\right\}$ (cf. for further details [81, Propostion 1.85]). The interpolation estimate for Raviart-Thomas element is stated in the following theorem.

Theorem 2.7. *Consider the interpolation operator $\Pi_\ell^{\mathrm{RT}} \colon V \to V_\ell^{\mathrm{RT}}$. For $\mathbf{v} \in V$ the interpolation estimate is given by*

$$\left\|\mathbf{v} - \Pi_\ell^{\mathrm{RT}}\mathbf{v}\right\|_{\mathrm{L}^2(D)} + \left\|\mathrm{div}(\mathbf{v} - \Pi_\ell^{\mathrm{RT}}\mathbf{v})\right\|_{\mathrm{L}^2(D)} \lesssim h\left(\|\mathbf{v}\|_{\mathrm{H}^1(D)} + \|\mathrm{div}\,\mathbf{v}\|_{\mathrm{H}^1(D)}\right),$$

where $V = \left\{\mathbf{v} \in \mathrm{H}^1(D, \mathbb{R}^{\mathbf{d}}) \colon \mathrm{div}\,\mathbf{v} \in \mathrm{H}^1(D)\right\}$.

Proof. Theorem 1.114 and corollary 1.115 in [81]. □

2.2.3 Discontinuous Galerkin Elements

The construction of the discontinuous Galerkin (dG) finite element space follows the same idea as of the Lagrange finite element space. Namely, the local basis functions are Lagrange polynomials and the dofs are given by point evaluations. However, the regularity requirement is much weaker, i.e., we only demand integrability of the solution over the domain and allow for discontinuities on the faces:

$$V_{\ell,\mathbf{p}\ge 0}^{\mathrm{dG}} = \begin{cases} \left\{v_\ell \in \mathrm{L}^2(D) \colon v_\ell|_K \in \mathbb{P}_{\mathbf{p}\ge 0},\ \forall K \in \mathcal{K}\right\}, & \text{for interval, triangle, tetrahedron} \\ \left\{v_\ell \in \mathrm{L}^2(D) \colon v_\ell|_K \in \mathbb{Q}_{\mathbf{p}\ge 0},\ \forall K \in \mathcal{K}\right\}, & \text{for quadrilateral, hexahedron, st-cell} \end{cases}$$

Discontinuous Galerkin methods have been getting popular for problems where the solution might form discontinuities, but they have also been applied to problems with rather smooth solutions [9]. We refer to [70, 134, 59] for a more detailed description of the theory and its applications. DG finite elements are closely related to finite volume methods, as they also require little regularity of the solution. For degree $\mathbf{p} = 0$, the methods coincide and the interpolation operator is given by

$$\Pi^{\mathrm{dG}}_{K,\mathbf{p}=0} \colon V(K) \ni v \longmapsto \left(\frac{1}{|K|} \int_K v \mathrm{d}\mathbf{x} \right) \mathbf{1}_K \in V^{\mathrm{dG}}_{K,\mathbf{p}=0} \quad \Rightarrow$$

$$\Pi^{\mathrm{dG}}_{\ell,\mathbf{p}=0} \colon V \ni v \longmapsto \sum_{K \in \mathcal{K}} \left(\frac{1}{|K|} \int_K v \mathrm{d}\mathbf{x} \right) \mathbf{1}_K \in V^{\mathrm{dG}}_{\ell,\mathbf{p}=0} \quad \text{with} \quad \mathbf{1}_K(\mathbf{x}) = \begin{cases} 1, & \mathbf{x} \in K \\ 0, & \mathbf{x} \notin K \end{cases}.$$

For higher degrees we take the local interpolation operator of Lagrange finite elements. The global interpolation operator on the other hand is given by

$$\Pi^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 1} \colon V \ni v \longmapsto \sum_{K \in \mathcal{K}} \sum_{\mathbf{z} \in \mathcal{Z}_K} v(\mathbf{z}) \psi_{K,\mathbf{z}} \in V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 1},$$

where the global set of nodal points is the non-overlapping union of the local nodal points. This usually leads to a significantly larger ansatz space compared to normal Lagrange finite elements, i.e., $\dim(V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 1}) \geq \dim(V^{\mathrm{Lag}}_{\ell,\mathbf{p}\geq 1})$. However, for some applications, especially for problems with dominant first order spatial derivatives like in electrodynamics, fluid mechanics, plasma physics and wave phenomena, the burden of the larger ansatz space has to be taken in order to cope with the low regularity. Additionally, the implementation of dG elements requires more communication across the processes since the distributed domains have to be overlapping. This is due to the fact that neighbouring cells have to be able to exchange information for this discretization.

Nevertheless, especially dG methods of high polynomial degree are popular to resolve for example vortices in aerodynamic flow simulations and to avoid strong damping (dissipation) of the discrete solution. Furthermore, dG methods provide a very easy way to develop hp-adaptivity, since the degree $\mathbf{p}_K$ and the mesh width h_K of each cell can be adapted independently and based on local error estimators [134].

Lemma 2.8. *For the dG-space we have* $V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 1} \subset \mathrm{H}^1(D)$, *if and only if* $[\![v]\!]_F = 0$. *Generally, this is a non-conforming (confer definition 2.15) space* $V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 1} \not\subset \mathrm{H}^1(D)$.

Proof. Confer [83, Lemma 1.75]. □

The previous lemma states that, if the jumps on the inner faces are non-zero, the ansatz space is non-conforming. To this end, the following definition is useful (confer [134, 9] or [81][section 3.2.4]):

Definition 2.9. We call $\mathrm{H}^k(\mathcal{K}_\ell) = \left\{ v \in \mathrm{L}^2(D) \colon v|_K \in \mathrm{H}^k(K), \forall K \in \mathcal{K}_\ell \right\}$ a *broken Sobolev space* on the triangulation. Let further $V(h_\ell) = V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0} + (\mathrm{H}^2(D) \cap \mathrm{H}^1_0(D)) \subset \mathrm{H}^2(\mathcal{K}_\ell)$ and define the following semi-norm and norm for $v_\ell \in V(h_\ell)$

$$\|v_\ell\|_{\mathrm{L}^2(\mathcal{K}_\ell)} = \sum_{K \in \mathcal{K}_\ell} \|v_\ell\|^2_{\mathrm{L}^2(K)}, \quad |v_\ell|^2_{\mathrm{H}^1(\mathcal{K}_\ell)} = \sum_{K \in \mathcal{K}_\ell} |v_\ell|^2_{\mathrm{H}^1(K)} = \sum_{K \in \mathcal{K}_\ell} \|\nabla v_\ell\|^2_{\mathrm{L}^2(K)}$$

Lastly, we want to remark that the local interpolation results of the Lagrange finite elements are still valid on this broken Sobolev space. However, since we have a globally non-conforming space, the final discretization scheme has to be carefully designed and the error estimation is significantly more technical.

2.2.4 Enriched Galerkin Elements

In [242, 163] another locally flux conservative finite element method as dG and RT elements is proposed. It is based on cell-wise constant enrichment of continuous Lagrange finite elements. The space is simply constructed by

$$V_{\ell,\mathbf{p}\geq 1}^{\mathrm{eG}} = V_{\ell,\mathbf{p}\geq 1}^{\mathrm{Lag}} + V_{\ell,\mathbf{p}=0}^{\mathrm{dG}}.$$

This space is locally and globally conservative such as the dG method, however, it requires significantly fewer degrees of freedom compared to the dG method which makes it very attractive for elliptic and parabolic problems. In [163] it also shown that it can be applied to cell-wise random permeabilities which makes this discretization especially interesting for the investigations in this thesis.

2.2.5 Space-Time Discontinuous Galerkin Elements

Lastly, we want to introduce space-time discontinuous Galerkin (ST-dG) elements. Basically, ST-dG elements are nothing else than dG elements defined on a space-time cylinder $Q := D \times (0,T)$, i.e.,

$$V_{\ell,\mathbf{p}_K,\mathbf{p}_I}^{\text{ST-dG}} = \left\{ \mathbf{v}_\ell \in \mathrm{L}^2((0,T),\mathrm{L}^2(D;\mathbb{R}^J)) \colon \mathbf{v}_\ell|_{K,I} \in \mathbb{Q}_{\mathbf{p}_K}(K;\mathbb{R}^J) \otimes \mathbb{P}_{\mathbf{p}_I}(I;\mathbb{R}^J),\ \forall K \in \mathcal{K}_\ell, \forall I \in \mathcal{I}_\ell \right\},$$

where $[0,T]$ is decomposed in disjoint time intervals $I \in \mathcal{I}$ of size τ_I with $[0,T] = \bigcup_{I\in\mathcal{I}} \overline{I}$. The space is discontinuous in space and time and suited for hyperbolic systems of conservation laws. For a more detailed description of the construction of the space-time cells, the resulting sceleton and the construction of the finite element space we refer to [62, section 3.1]. Furthermore, it does provide a natural **p**-adaptivity in space and time guided by local error estimates [62, section 6]. There are also spaces which use a discontinuous Galerkin scheme in space and a Petrov-Galerkin scheme in time [75, 83] which saves degrees of freedom for the time discretization. These methods have been popularized, due to the fact that they are capable to parallelize in space and time. Thereby, they can unfold their potential on large computers with parallel multigrid preconditioning [77, 269]. Further details will follow in the next section, or can be found in [76, 23, 207] among many other sources.

2.2.6 Multi-sample Finite Elements

We recall the developed parallelization of the triangulation in section 2.1 and motivate with that the definition of finite element spaces, parallelized over the input data $\{\mathbf{y}^{(m)}\}_{m=1}^{M_\ell}$.

Definition 2.10. We call the space $V_{\ell,\mathcal{P}} = V_\ell^{(1)} \times \cdots \times V_\ell^{(M_\ell)} = \prod_{m=1}^{M_\ell} V_\ell^{(m)}$ a *multi-sample finite element* *space*, where

$$V_\ell^{(m)} := \left\{ \mathbf{v}_\ell \in V_\ell(D;\mathbb{R}^J) \colon \mathbf{v}_\ell|_K \in V_\ell(K;\mathbb{R}^J),\, \forall K \in \mathcal{K}^{(m)}_{\ell,\mathcal{P}_k^{(m)}},\, \mathcal{P}_k^{(m)} \subset \mathcal{P} \right\}$$

is an arbitrary finite element space for a single sample, parallelized in space on a subdomain of processes $\mathcal{P}_k \subset \mathcal{P}$ where k is chosen according to (2.4).

Remark 2.11. a) The space $V_{\ell,\mathcal{P}}$ will later in section 2.4 be used to search the approximative solution to multiple samples. By considering the available processes $\mathcal{P}$ in the ansatz space, this enables to construct an adaptive parallelization scheme and a data structure spanning over the spatial and the parametric domain.

b) This idea is applicable to arbitrary FE-spaces, thus to all spaces which were introduced here in section 2.2, as well as other finite elements e.g. as listed in section 2.5.

2.3 Finite Element Approximation of Weak Solutions

As we will see later in this section, we can derive *weak* formulations for the model problems. To this end, we introduce the following abstract problem.

Model Problem 2.12. Let $(V, \|\cdot\|_V)$ and $(\tilde{V}, \|\cdot\|_{\tilde{V}})$ be Banach spaces with $\tilde{V}$ being reflexive. The abstract problem reads as: Seek $u \in V$, such that

$$a(u,v) = f(v) \quad \text{for all} \quad v \in \tilde{V}. \tag{2.8}$$

Here, $a\colon V \times \tilde{V} \to \mathbb{R}$ is a continuous bilinear form and $f\colon \tilde{V} \to \mathbb{R}$ is a continuous linear form. Often problems in this form are called *variational* or *weak* formulations.

A bilinear form as stated in the above problem can be derived for the problems 1.1-1.3. The linearity in the first argument is a result of the linearity of the here considered model problems, whereas the linearity in the second argument follows from the construction of weak formulations. For existence theory on this abstract problem and thereby, existence theory of solutions to the deterministic PDEs, we refer to the appendix A.1.

Definition 2.13. The problem (2.8) is called *well-posed*, if it admits a *unique solution* $u \in V$ and if the solution is controlled by the right-hand side, i.e., $\|u_f\|_V \lesssim \|f\|_{\tilde{V}'}$ for any fixed $f \in \tilde{V}'$ with $\tilde{V}'$ being the dual space to $\tilde{V}$.

Now, we replace the infinite dimensional function spaces V and $\tilde{V}$ by finite dimensional finite element spaces as introduced in the previous section. The problem then reads as:

Model Problem 2.14. Let V_ℓ and $\tilde{V}_\ell$ be finite element spaces. The discrete template problem reads as, seek $u_\ell \in V_\ell$, such that

$$a_\ell(u_\ell, v_\ell) = f_\ell(v_\ell) \quad \text{for all} \quad v_\ell \in \tilde{V}_\ell. \tag{2.9}$$

Here, $a_\ell\colon V \times \tilde{V} \to \mathbb{R}$ is a continuous bilinear form and $f_\ell\colon \tilde{V} \to \mathbb{R}$ is a continuous linear form. Often problems in this form are called *Galerkin* approximations, if $V_\ell = \tilde{V}_\ell$ and *Petrov-Galerkin* approximations, if $V_\ell \neq \tilde{V}_\ell$.

To validate the (Petrov)-Galerkin approximations introduced in this section, we recall the properties of a discretization method.

Definition 2.15. a) A discretization method is called *conform*, if $V_\ell \subset V$ and $\tilde{V}_\ell \subset \tilde{V}$.

b) A discretization method is called *consistent*, if the true solution $u \in V$ to (2.8) solves the discrete system (2.9), i.e. $a_\ell(u, v) = f_\ell(v)$ for all $v \in \tilde{V}$.

c) The discretization error $u - u_\ell$ satisfies the *Galerkin orthogonality*, if it is orthogonal to the test space $\tilde{V}_\ell$ with respect to the induced inner product by the bilinear form, i.e., $a_\ell(u - u_\ell, v_\ell) = 0$ for all $v_\ell \in \tilde{V}_\ell$. This is equivalent to consistency by subtracting (2.9) from (2.8).

d) A discretization is called *coercive*, if there exists a constant, such that $a_\ell(v_\ell, v_\ell) \gtrsim \|v_\ell\|_V^2$ for all $v_\ell \in V_\ell \subset V$ (confer definition A.4 b)).

e) If the discretization method is coercive and the right-hand side of (2.9), i.e., the linear form $f\colon \tilde{V} \to \mathbb{R}$, is bounded, the discretization is *stable*, since for the solution $u_\ell \in V_\ell$ to (2.9), we have

$$\|u_\ell\|_V^2 \lesssim a_\ell(u_\ell, u_\ell) = f_\ell(u_\ell) \lesssim \|u_\ell\|_V\,,$$

which gives control over the discrete solution by $\|u_\ell\|_V \lesssim 1$.

f) Given the true solution to (2.8) by $u \in V$, the approximation $u_\ell \in V$ from (2.9) and an appropriate norm $\|\cdot\|_V$ to measure the error, a discretization method is called *convergent* of order α, if

$$\mathrm{err}_{\mathrm{disc}} \coloneqq \|u - u_\ell\|_V \lesssim h_\ell^\alpha \quad \Rightarrow \quad \lim_{h_\ell \to 0} \|u - u_\ell\|_V = 0,$$

where the hidden constant is dependent on the regularity of the true solution.

g) Given a bounded quantity of interest (QoI) $\mathrm{Q}\colon V \to \mathbb{R}$ of the solution, convergence w.r.t. this quantity is expressed by (1.2) and 2.15 f) via

$$|\mathrm{Q}(u) - \mathrm{Q}(u_\ell)| \lesssim h_\ell^\alpha \|u\|_V\,.$$

h) An error estimate is called *a priori*, if it depends on measures of the true solution. Hence, a priori estimates can not be used to quantify the size of the error. They are used to predict the order of convergence of a method applied to a problem prior to the actual computation.

i) An error estimates is called *a posteriori*, if it only depends on quantifiable values, hence they can be used as an error estimate after a computation has been performed to validate the result.

In the following, we will revisit the introduced model problems from section 1.3, neglecting the dependency on the uncertainty for now. We will follow [81, 134] and state the setting of Galerkin methods. The goal is to derive for every model problem the (deterministic) discrete weak formulation of the system. We mostly remained by the notation used in [28].

The Galerkin Approximation

We consider again the abstract problem 2.12 and its discretized version 2.14. For this problem, we can state the following.

Theorem 2.16. *Let $V_\ell \subset V$ with $\dim V_\ell < \infty$. Let $a\colon V \times V \to \mathbb{R}$ be a symmetric and positive definite bilinear form and let $f : V \to \mathbb{R}$ be a linear form. Then model problem 2.14 has a unique solution.*

Proof. Let $\{\psi_1, \ldots, \psi_N\}$ be the basis of V_ℓ with $\dim V_\ell = N$. Plugging $u_\ell = \sum_{n=1}^N \mu_n \psi_n \in V_\ell$ and $v_\ell = \sum_{n=1}^N \nu_n \psi_n \in V_\ell$ into 2.14 and reformulating it as an algebraic system gives

$$\boldsymbol{\nu}^\top \underline{A}\boldsymbol{\mu} = \boldsymbol{\nu}^\top \mathbf{b} \quad \forall \boldsymbol{\nu} \in \mathbb{R}^N \quad \Rightarrow \quad \underline{A}\boldsymbol{\mu} = \mathbf{b},$$

with $\underline{A} = (a(\psi_i, \psi_j))_{i,j=1}^N$, $\mathbf{b} = [f(\psi_1), \ldots, f(\varphi_N)]^\top$ and $\boldsymbol{\mu}, \boldsymbol{\nu} \in \mathbb{R}^N$. Since $a\colon V \times V \to \mathbb{R}$ is symmetric and positive definite, the matrix $\underline{A}$ is symmetric and positive definite and thereby invertible, giving access to a unique solution $u_\ell = \sum_{n=1}^N \mu_n \psi_n \in V_\ell$. □

Remark 2.17. a) The proof of theorem 2.16 already demonstrates the assembling of the finite element system. We choose a finite dimensional function space with known basis functions and plug the linear combinations of u_ℓ and v_ℓ into the weak discrete formulation, leading to a linear system. By solving this linear system, we have the coefficients representing the finite element solution.

b) Since these methods arose from mechanical engineering applications, the matrix $\underline{A}$ is traditionally called *stiffness matrix* and the vector b is called *load vector*. Further details on this linear equation system, how to assemble it and how to solve it is provided in the section 2.4.

To quantify how good the Galerkin approximation is, we state the following essential lemma. It can be found in any book about FEM such as [42, 44, 125].

Lemma 2.18 (Céa's Lemma)**.** *Let $a\colon V \times V \to \mathbb{R}$ be a continuous, symmetric and coercive bilinear form. If $u \in V$ is a solution of (2.8) and $u_\ell \in V_\ell$ is the solution of (2.9) with $V_\ell \subset V$, then, with the norm $\|v\|_a^2 = a(v, v)$, it holds*

$$\|u_\ell - u\|_a \leq \inf_{v_\ell \in V_\ell} \|v_\ell - u\|_a .$$

Proof. Since $V_\ell \subset V$ and (2.8) holds for any $v \in V$, it is $a(u, v_\ell) = f(v_\ell)$ for all $v_\ell \in V_\ell$. Subtracting this from (2.9) gives $a(u_\ell - u, v_\ell) = 0$ for all $v_\ell \in V_\ell$. By choosing v_ℓ as $u_\ell - v_\ell \in V_\ell$, we get $a(u_\ell - u, u_\ell - v_\ell) = 0$ for all $v_\ell \in V_\ell$. Furthermore, inserting $-u + u$ into the second argument and rearranging the equation, i.e., $a(u_\ell - u, u_\ell - u) = a(u_\ell - u, v_\ell - u)$, gives with the Cauchy-Schwarz inequality

$$\|u_\ell - u\|_a^2 = a(u_\ell - u, u_\ell - u) = a(u_\ell - u, v_\ell - u) \leq \|u_\ell - u\|_a \|v_\ell - u\|_a , \quad \forall v_\ell \in V_\ell.$$

The result follows by division of $\|u_\ell - u\|_a$ and taking the infimum over all $v_\ell \in V_\ell$. □

Remark 2.19. Céa's lemma states that the Galerkin approximation $u_\ell \in V_\ell$ is the optimal approximation of the solution $u \in V$ with respect to the energy norm. Thus, if we want to approximate the solution u better, we have to approximate the space V by V_ℓ better which motivates us to revisit the spaces of section 2.2 within the scope of the model problems 1.1-1.3.

2.3.1 Elliptic Subsurface Diffusion Problem

Starting with the deterministic version of model problem 1.1 by neglecting the dependency on $\omega \in \Omega$ with $f \in \mathrm{L}^2(D)$, $u_D \in \mathrm{L}^2(\Gamma_D)$ and $g_N \in \mathrm{L}^2(\Gamma_N)$ as input data, we derive the weak formulation by multiplying with a test function $v \in \mathrm{H}^1_0(D)$, integrating over the whole domain D and applying integration by parts. This yields a weak formulation in the form (2.12) with a symmetric, continuous and coercive bilinear form. By the Lax-Milgram theorem A.5, this problem with homogeneous boundary data admits a unique solution $u \in \mathrm{H}^1_0(D)$ and thereby is well-posed by definition 2.13. Additional regularity can be achieved with sufficient input data and with the trace theorem [42, chapter 3] non-homogeneous boundary data can be incorporated. We omit details and refer to [42] or [134, section 2.1].

Main intent of this paragraph is to introduce four different discretization methods to model problem 1.1, starting with a conforming Galerkin approximation with Lagrange finite elements.

Model Problem 2.20. We search for an approximative solution $u_\ell \in V^{\mathrm{Lag}}_{\ell,\mathbf{p}\geq 1}(u_{\mathrm{D}})$ using Lagrange elements solving

$$\underbrace{\int_D \kappa \nabla u_\ell \cdot \nabla v_\ell \,\mathrm{d}\mathbf{x}}_{=:a^{\mathrm{Lag}}_\ell(u_\ell, v_\ell)} = \underbrace{\int_D f_\ell v_\ell \,\mathrm{d}\mathbf{x} + \int_{\Gamma_{\mathrm{N}}} g_{\mathrm{N}} v_\ell \,\mathrm{d}a}_{=:b^{\mathrm{Lag}}_\ell(v_\ell)}, \quad \forall v_\ell \in V^{\mathrm{Lag}}_{\ell,\mathbf{p}\geq 1}(0)$$

with $V^{\mathrm{Lag}}_{\ell,\mathbf{p}\geq 1}(u_{\mathrm{D}}) = \left\{ u_\ell \in V^{\mathrm{Lag}}_{\ell,\mathbf{p}\geq 1} : u_\ell(\mathbf{z}) = u_{\mathrm{D}}(\mathbf{z}),\, \mathbf{z} \in \mathcal{Z} \cap \Gamma_{\mathrm{D}} \right\}$ to satisfy the Dirichlet boundary conditions which are called *essential*, since they are considered directly in the ansatz space.

Proposition 2.21. *a) The Lagrange discretization is conform since* $V^{\mathrm{Lag}}_{\ell,\mathbf{p}\geq 1} \subset \mathrm{H}^{\min\{s,\mathbf{p}\}+1}(D)$.

b) The Lagrange discretization is consistent since $a^{\mathrm{Lag}}_\ell(\cdot,\cdot) \equiv a(\cdot,\cdot)$ *and* $b^{\mathrm{Lag}}_\ell(\cdot) \equiv b(\cdot)$.

c) The Lagrange discretization is coercive w.r.t. to the induced norm of the bilinear form $\|\cdot\|_a$ *and w.r.t.* $\|\cdot\|_{\mathrm{H}^1(D)}$.

d) A solution to the problem exists and is unique.

e) The Lagrange discretization is convergent for $u \in \mathrm{H}^{\min\{s,\mathbf{p}\}+1}(D)$ *and* $u_\ell \in V^{\mathrm{Lag}}_{\ell,\mathbf{p}\geq 1}$

$$\|u_\ell - u\|_{\mathrm{H}^1(D)} \lesssim h_\ell^{\min\{s,\mathbf{p}\}} |u|_{\mathrm{H}^{\min\{s,\mathbf{p}\}+1}(D)} \quad \text{and} \quad \|u_\ell - u\|_{\mathrm{L}^2(D)} \lesssim h_\ell^{\min\{s,\mathbf{p}\}+1} |u|_{\mathrm{H}^{\min\{s,\mathbf{p}\}+1}(D)}.$$

Proof. These results are derived by combining Céa's lemma 2.18 and the interpolation estimate 2.5, using theorem 2.16 and applying the Aubin-Nitsche trick. We refer to standard literature e.g. [44, 81] for details. □

Often enough we are interested in the flux $\mathbf{q} = -\kappa \nabla u$ which is also assumed to be divergence free $\operatorname{div} \mathbf{q} = 0$. To this end, $\operatorname{div} \mathbf{q} = 0$ and $\mathbf{q} = -\kappa \nabla u$ are both tested with sufficiently smooth functions $v \in C^\infty_0(D)$ and $\boldsymbol{\psi} \in C^\infty_0(D, \mathbb{R}^{\mathbf{d}})$ whereby a mixed weak formulation can be derived. To enforce physically correct flux fields (flux preserving discretizations), mixed FEM are popular, where we discretize the pressure head u and the flux $\mathbf{q}$ in separated spaces [46].

Model Problem 2.22. Using mixed finite elements, we search for the approximative solution $(\mathbf{q}_\ell, u_\ell) \in V_\ell^{\mathrm{RT}}(-g_N) \times V_{\ell,\mathbf{p}=0}^{\mathrm{dG}}$, such that

$$\int_D \kappa^{-1}\mathbf{q}_\ell \cdot \boldsymbol{\psi}_\ell \,\mathrm{d}\mathbf{x} - \int_D u_\ell \operatorname{div} \boldsymbol{\psi}_\ell \,\mathrm{d}\mathbf{x} = -\int_{\Gamma_\mathrm{D}} u_\mathrm{D} \boldsymbol{\psi}_\ell \cdot \mathbf{n} \,\mathrm{d}a$$

$$\int_D \operatorname{div} \mathbf{q}_\ell v_\ell \,\mathrm{d}\mathbf{x} = 0, \quad \forall (\boldsymbol{\psi}_\ell, v_\ell) \in V_\ell^{\mathrm{RT}}(0) \times V_{\ell,\mathbf{p}=0}^{\mathrm{dG}},$$

where the Neumann data is imposed as the essential boundary conditions by

$$V_\ell^{\mathrm{RT}}(-g_N) = \left\{ \mathbf{v}_\ell \in V_\ell^{\mathrm{RT}} \colon \int_F \mathbf{v}_\ell \cdot \mathbf{n} \,\mathrm{d}a = -\int_F g_N \,\mathrm{d}a \text{ for } F \in \mathcal{F},\, \mathbf{z}_F \in \Gamma_N \right\}.$$

The above formulation is a saddle point problem which often appear in optimization problems under a constraint. This motivates an equivalent formulation which can be derived by introducing an additional space defined on the element boundaries corresponding to a Lagrange multiplier. This technique is commonly known as hybridization and allows to reduce the global problem to a symmetric positive definite system (confer [46, chapter V] or [38]). This system can be solved independently for each cell and thereby can be parallelized very well. The actual solution $(\mathbf{q}_\ell, u_\ell)$ is reconstructed by a post-processing step. The downside of this approach is that the system is also significantly larger than for the pure mixed formulation. We shortly discuss this again in the numerical results in section 4.1.

Model Problem 2.23. Introducing a Lagrange multiplier space $G_\ell = \prod_{F \in \mathcal{F}} \mathbb{P}_0(F)$ for the element boundary flux and the discontinuous space $V_\mathcal{K} = \prod_{K \in \mathcal{K}} V_\ell^{\mathrm{RT}}(K)$, the mixed approximation can be computed by $(\mathbf{q}_\ell, u_\ell, \lambda_\ell) \in V_\mathcal{K} \times V_{\ell,\mathbf{p}=0}^{\mathrm{dG}} \times G_\ell(-u_\mathrm{D})$ solving the extended saddle point problem

$$\begin{aligned}
\int_K \kappa^{-1}\mathbf{q}_\ell \cdot \boldsymbol{\psi}_\ell \,\mathrm{d}\mathbf{x} - \int_K u_\ell \operatorname{div} \boldsymbol{\psi}_\ell \,\mathrm{d}\mathbf{x} &= \int_{\partial K} \lambda_\ell \boldsymbol{\psi}_K \cdot \mathbf{n} \,\mathrm{d}a\,, && \boldsymbol{\psi}_K \in V_\ell^{\mathrm{RT}}(K)\,,\\
\int_K \operatorname{div} \mathbf{q}_\ell v_\ell \,\mathrm{d}x &= 0\,, && v_\ell \in V_{\ell,\mathbf{p}=0}^{\mathrm{dG}}(K),\\
\sum_{K \in \mathcal{K}} \int_{\partial K} \mathbf{q}_\ell \cdot \mathbf{n}\, \mu_\ell \,\mathrm{d}a &= -\int_{\Gamma_\mathrm{N}} g_\mathrm{N}\, \mu_\ell \,\mathrm{d}a\,, && \mu_\ell \in G_\ell(0)
\end{aligned}$$

with $G_\ell(-u_\mathrm{D}) = \left\{ \mu_\ell \in G_\ell \colon \int_F \mu_\ell \,\mathrm{d}a = -\int_F u_\mathrm{D} \,\mathrm{d}a\,, F \in \mathcal{F} \cap \Gamma_\mathrm{D} \right\}$. Therefore, the Dirichlet boundary conditions are essential again and included in the Lagrange multiplier space.

Since model 2.22 and model 2.23 are equivalent, the following proposition holds for both discretization approaches. We follow [38] and state the following proposition.

Proposition 2.24. *a) Let $(\mathbf{q}, u) \in \mathrm{H}_{\Gamma_N}(\operatorname{div}, D) \times \mathrm{L}^2(D)$ be the solution to the exact saddle point problem (confer [38, problem 7.1.17]) and let $(\mathbf{q}_\ell, u_\ell) \in V_\ell^{\mathrm{RT}}(-g_N) \times V_{\ell,\mathbf{p}=0}^{\mathrm{dG}}$ be the solution to model 2.22. Then* *the error can be estimated for $s \leq 1$ with*

$$\|\mathbf{q} - \mathbf{q}_\ell\|_{\mathrm{L}^2(D)} \lesssim h_\ell^{s+1} \|\mathbf{q}\|_{\mathrm{H}^{s+1}(D)} \quad \text{and} \quad \|u - u_\ell\|_{\mathrm{L}^2(D)} \lesssim h_\ell^{s+1}(\|u\|_{\mathrm{H}^{s+1}(D)} + \|\mathbf{q}\|_{\mathrm{H}^{s+1}(D)})$$

if u and $\mathbf{q}$ provide the sufficient regularity.

Proof. Can be found in [38, proposition 7.1.2]. □

Lastly, we consider the elliptic problem discretized with discontinuous Galerkin finite elements. The resulting discretization has more degrees of freedom than the standard Lagrange discretization but offers flux preservation as well. Furthermore, diffusion terms such as in elliptic model problem 3.23 also appear in other PDEs which might be discretized with dG methods. Hence, developing a dG discretization for this particular term can be employed in other PDEs as well.

Model Problem 2.25. Using the discontinuous Galerkin (dG) method with the penalty parameter $\gamma = \mathcal{O}(\mathbf{p}^2)$, we search for a solution $u_\ell \in V_{\ell,\mathbf{p}\geq 0}^{\mathrm{dG}}$ satisfying

$$a_\ell^{\mathrm{dG}}(u_\ell, v_\ell) = b_\ell^{\mathrm{dG}}(v_\ell), \quad \forall v_\ell \in V_{\ell,\mathbf{p}\geq 0}^{\mathrm{dG}},$$

where for $u_\ell, v_\ell \in V_{\ell,p}^{\mathrm{dG}}$ the bilinear form and the linear form are given by

$$\begin{aligned} a_\ell^{\mathrm{dG}}(u_\ell, v_\ell) &= \sum_{K\in\mathcal{K}} \int_K \kappa \nabla u_\ell \cdot \nabla v_\ell \,\mathrm{d}\mathbf{x} + \sum_{F\in\mathcal{F}\setminus\Gamma_\mathrm{N}} \frac{\gamma}{h_F} \int_F [\![u_\ell]\!]_F \cdot [\![v_\ell]\!]_F \,\mathrm{d}a \\ &\quad - \sum_{F\in\mathcal{F}\setminus\Gamma_\mathrm{N}} \int_F (\{\!\{\kappa\nabla u_\ell\}\!\}_F \cdot [\![v_\ell]\!]_F + [\![u_\ell]\!]_F \cdot \{\!\{\kappa\nabla v_\ell\}\!\}_F) \,\mathrm{d}a, \\ b_\ell^{\mathrm{dG}}(v_\ell) &= \int_D f v_\ell \mathrm{d}\mathbf{x} + \int_{\Gamma_\mathrm{N}} g_\mathrm{N} v_\ell \,\mathrm{d}a + \sum_{F\in\mathcal{F}\cap\Gamma_\mathrm{D}} \frac{\gamma}{h_F} \int_F u_\mathrm{D} v_\ell \,\mathrm{d}a \\ &\quad - \sum_{F\in\mathcal{F}\cap\Gamma_\mathrm{D}} \int_F u_\mathrm{D} \kappa \nabla v_\ell \cdot \mathbf{n} \,\mathrm{d}a, \end{aligned}$$

with $[\![\cdot]\!]$ as face jump and $\{\!\{\cdot\}\!\}$ as face average (confer (2.7)). We refer to [9] for more details.

Proposition 2.26. *a) The dG discretization is non-conforming since $V_{\ell,\mathbf{p}\geq 0}^{\mathrm{dG}} \not\subset \mathrm{H}^1(D)$.*

b) The bilinear form is symmetric, i.e., $a_\ell^{\mathrm{dG}}(u_\ell, v_\ell) = a_\ell^{\mathrm{dG}}(v_\ell, u_\ell)$.

c) The dG discretization is consistent, i.e., the error to the solution $u \in \mathrm{H}^2(D) \cap \mathrm{H}_0^1(D)$ satisfies the Galerkin orthogonality $a_\ell^{\mathrm{dG}}(u - u_\ell, v_\ell) = 0$ for all $v_\ell \in V_{\ell,\mathbf{p}\geq 0}^{\mathrm{dG}}$.

d) The dG discretization is coercive, i.e., $a_\ell(u_\ell, u_\ell) \gtrsim |\!|\!| u_\ell |\!|\!|^2$ with respect to the norm

$$|\!|\!| u_\ell |\!|\!|^2 := \|\nabla u_\ell\|_{\mathrm{L}^2(\mathcal{K}_\ell)}^2 + \sum_{F\in\mathcal{F}\setminus\Gamma_N} \tfrac{1}{h_F} \|[\![u_\ell]\!]_F\|_{\mathrm{L}^2(F)}^2 .$$

e) The dG bilinear form is bounded, i.e., for $u_\ell \in V(h_\ell)$ (confer definition 2.9)

$$a_\ell^{\mathrm{dG}}(u_\ell, v_\ell) \lesssim |||u_\ell||| \cdot |||v_\ell|||$$

where $|||u_\ell|||^2 = \|u_\ell\|^2 + \sum_{F\in\mathcal{F}\setminus\Gamma_\mathrm{N}} h_F \left\|\nabla u_\ell \cdot \mathbf{n}_F\right\|^2_{\mathrm{L}^2(F)}$.

f) It is $\|u - u_\ell\| \lesssim \inf_{v_\ell\in V} |||u - v_\ell|||$.

g) For $u \in \mathrm{H}^{s+1}(D)$ *the error can be estimated with* $\|u - u_\ell\| \lesssim h_\ell^s \|u\|_{\mathrm{H}^{s+1}(D)}$.

h) For $u \in \mathrm{H}^{s+1}(D)$ *the error can be estimated with* $\|u - u_\ell\|_{\mathrm{L}^2(D)} \lesssim h_\ell^{s+1} \|u\|_{\mathrm{H}^{s+1}(D)}$.

Proof. a) Follows by lemma 2.8, b) follows by the construction of the bilinear form with *symmetric inner penalty* and c) is given since for $u \in H^2(D)$ the jump $[\![u]\!]_F$ and the face average $\{\!\{\nabla u\}\!\}_F \cdot \mathbf{n}_K$ vanish for faces $F \in \mathcal{F} \subset \Gamma_N$. The coercivety in d) and the boundedness in e) of the bilinear form can be found in [9, section 4.1 and 4.2] or [134, section 5.4] and the convergence f), g) and h) in [9, section 5] or in [134, section 5.5]. □

Remark 2.27. The usage of enriched Galerkin (eG) elements can be done with the same discrete weak formulation as in model 2.25 to get again cell-wise flux preservation. We omit the details and refer to [163, 163]. However, the ansatz space for eG elements has fewer degrees of freedom. Thus, the method is cheaper than conventional dG methods. We will investigate this in section 4.1 in further detail.

2.3.2 Linear Hyperbolic Conservation Laws

Before we move to the next model problem, we shortly review the structure of linear hyperbolic conservation laws of first-order. These problems often arise in continuum physics and solutions to these problems are space and time dependent, informally described as *wave-like*. We refer to standard literature [84, chapter 7.3] and [166, 115] on the topic, but will mostly follow [141] and [76] because our particular application cases are covered in these sources and a similar notation to the one presented here is used. Generally, these systems are of the form: search for a collection of quantities $\mathbf{u}(t) \in V \subset \mathrm{L}^2(D, \mathbb{R}^J)$, such that

$$M\partial_t \mathbf{u}(t) + A\mathbf{u}(t) = \mathbf{b}(t) \quad \text{for} \quad t \in (0, T] \quad \text{and} \quad \mathbf{u}(0) = \mathbf{u}_0, \tag{2.10}$$

subject to some boundary conditions, e.g., sufficient homogeneous Dirichlet boundary conditions. For the solution space V we consider a Hilbert space with weighted inner product $\langle \mathbf{v}, \boldsymbol{\phi}\rangle_V = \langle M\mathbf{v}, \boldsymbol{\phi}\rangle_{\mathrm{L}^2(D)}$, where $M \in \mathrm{L}^\infty(D, \mathbb{R}^{J\times J}_{\mathrm{sym}})$ is an uniformly positive mass operator, i.e., $\langle M\mathbf{v}, \mathbf{v}\rangle_{L^2(D)} \gtrsim \|\mathbf{v}\|^2_{L^2(D)} > 0$ for all $0 \neq \mathbf{v} \in \mathrm{L}^2(D, \mathbb{R}^J)$.

The operator A is a first order differential operator in space and linear on V and we assume that we can express it with the symmetric matrices $\underline{B}_d \in \mathbb{R}^{J\times J}_{\mathrm{sym}}$ as

$$\mathrm{L}^2(D, \mathbb{R}^J) \ni A\mathbf{u} = \sum_{d=1}^{\mathbf{d}} \partial_d(\underline{B}_d \mathbf{u}) = \sum_{d=1}^{\mathbf{d}} \underline{B}_d(\partial_d \mathbf{u}), \quad \mathbf{u} \in \mathcal{D}(A) \subset \mathrm{L}^2(D, \mathbb{R}^J),$$

where we denote with $\mathcal{D}(A) \subset V$ the domain of A. With integration by parts, sufficiently smooth test functions vanishing at the boundary and by exploiting the symmetry of $\underline{B}_d$, we see

$$\langle A\mathbf{u}, \boldsymbol{\phi}\rangle_{\mathrm{L}^2(D)} = \sum_{d=1}^{\mathbf{d}} \int_D \underline{B}_d \partial_d \mathbf{u} \cdot \boldsymbol{\phi} \,\mathrm{d}\mathbf{x} = -\sum_{d=1}^{\mathbf{d}} \int_D \mathbf{u} \cdot \underline{B}_d \partial_d \boldsymbol{\phi} \,\mathrm{d}\mathbf{x} = -\langle \mathbf{u}, A\boldsymbol{\phi}\rangle_{\mathrm{L}^2(D)} .$$

Hence, the operator is skew-symmetric $A^* = -A$. To incorporate boundary conditions, we additionally define with $A_{\mathbf{n}} = \sum_{d=1}^{\mathbf{d}} n_d \underline{B}_d^{\mathbf{n}}$ for $\mathbf{n} \in \mathbb{R}^d$ and $\underline{B}_d^{\mathbf{n}} \in \mathbb{R}_{\mathrm{sym}}^{J\times J}$ the boundary flux operator. Furthermore, if $\mathbf{b} \equiv 0$, the energy $\mathrm{E}(\mathbf{v}) := \frac{1}{2}\langle \mathbf{v}, \mathbf{v}\rangle_V$ is conserved, i.e., for $\mathbf{u}(t) \in \mathcal{D}(A)$

$$\partial_t \mathrm{E}(\mathbf{u}(t)) = \langle M\partial_t \mathbf{u}(t), \mathbf{u}(t)\rangle_{\mathrm{L}^2(D)} = -\langle A\mathbf{u}(t), \mathbf{u}(t)\rangle_{\mathrm{L}^2(D)} = 0$$

is satisfied. With the matrices $\underline{B}_1, \dots, \underline{B}_{\mathbf{d}} \in \mathbb{R}_{\mathrm{sym}}^{J\times J}$ we can rewrite (2.10) with

$$\operatorname{div} \mathbf{F}(\mathbf{u}) := \sum_{d=1}^{\mathbf{d}} \underline{B}_d(\partial_d \mathbf{u}) = A\mathbf{u}.$$

Definition 2.28. We call a problem: seek $\mathbf{u}(t) \in V \subset \mathrm{L}^2(D)$ such that

$$M\partial_t \mathbf{u}(t) + \operatorname{div} \mathbf{F}(\mathbf{u}(t)) = \mathbf{b}(t) \quad \text{for} \quad t \in (0,T] \quad \text{and} \quad \mathbf{u}(0) = \mathbf{u}_0, \tag{2.11}$$

subject to some boundary conditions a *conservation law* if $\mathbf{b} \equiv \mathbf{0}$ or a *balance law* if $\mathbf{b} \not\equiv \mathbf{0}$.

Definition 2.29. A system of the form (2.11) is called *hyperbolic*, if for any direction $\mathbf{n} = (n_1, \dots, n_{\mathbf{d}})^\top \in \mathbb{R}^{\mathbf{d}}$, $|\mathbf{n}| = 1$ the matrix

$$\underline{B} = \sum_{d=1}^{\mathbf{d}} n_d \underline{B}_d, \qquad \underline{B}_d \in \mathbb{R}_{\mathrm{sym}}^{J\times J} \tag{2.12}$$

has J real eigenvalues $\lambda_1(\mathbf{n}) \le \lambda_2(\mathbf{n}) \le \dots \le \lambda_J(\mathbf{n})$ with a complete family of eigenvectors, i.e., the system is diagonalizable.

Definition 2.30. We call a function $\mathbf{u} \in \mathrm{L}^2((0,T)\times D, \mathbb{R}^J)$ a *weak solution* to (2.11), if

$$\int_D M(\mathbf{x})\mathbf{u}_0(\mathbf{x})\cdot\boldsymbol{\phi}(\mathbf{x},0)\,\mathrm{d}\mathbf{x} + \int_{(0,T)\times D} \big(M(\mathbf{x})\mathbf{u}(\mathbf{x},t)\cdot\partial_t\boldsymbol{\phi}(\mathbf{x},t) + \mathbf{F}(\mathbf{u}(\mathbf{x},t))\cdot\nabla\boldsymbol{\phi}(\mathbf{x},t)\big)\,\mathrm{d}t\,\mathrm{d}\mathbf{x}$$
$$= \int_{(0,T)\times D} \mathbf{b}(\mathbf{x},t)\boldsymbol{\phi}(\mathbf{x},t)\mathrm{d}t\,\mathrm{d}\mathbf{x}$$

for all test functions $\boldsymbol{\phi} \in \{\mathbf{v} \in C^1((0,T)\times D, \mathbb{R}^J) \colon \mathbf{v}|_{\partial D} = 0 \text{ and } \mathbf{v}(\cdot,T) = 0\}$.

All solutions to (2.11) are also weak solutions, however, the opposite can not be implied. For further details on well-posedness of the problem we refer to [141]. Based upon the formulation (2.11) the numerical methods are constructed.

Semi-Discretization of Hyperbolic Conservation Laws. Following [75, 141], we will now outline the semi-discretization of an arbitrary hyperbolic conservation law with a discontinuous Galerkin approximation. After that we present the particular expression of the discretization for the transport model and the acoustic wave equation. We consider (2.11) and develop a system which is discretized in space but not in time yet, i.e., we search for $\mathbf{u}_\ell \in V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0}$ using dG elements, such that

$$\underline{M}_\ell \partial_t \mathbf{u}_\ell(t) + \underline{A}_\ell \mathbf{u}_\ell(t) = \mathbf{b}_\ell(t), \qquad t \in (0,T) \tag{2.13}$$

with $\mathbf{b}_\ell \in \mathrm{L}^2((0,T), V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0})$ and with the discrete operator $\underline{M}_\ell \in \mathcal{L}(V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0}, V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0})$ as a block diagonal and positive definite matrix (restricted to cell-wise constant material parameters). This operator is determined by a Galerkin approximation, i.e.,

$$\langle \underline{M}_\ell \mathbf{u}_\ell, \boldsymbol{\phi}_\ell \rangle_{\mathrm{L}^2(D)} = \langle M \mathbf{u}_\ell, \boldsymbol{\phi}_\ell \rangle_{\mathrm{L}^2(D)} \quad \text{and} \quad \langle \mathbf{b}_\ell, \boldsymbol{\phi}_\ell \rangle_{\mathrm{L}^2(D)} = \langle \mathbf{b}, \boldsymbol{\phi}_\ell \rangle_{\mathrm{L}^2(D)}, \qquad \mathbf{u}_\ell, \boldsymbol{\phi}_\ell \in V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0}.$$

To construct the operator $\underline{A}_\ell \in \mathcal{L}(V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0}, V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0})$ we consider once again (2.11). Then, with sufficiently smooth test functions restricted to a single cell $K \in \mathcal{K}$ and with integration by parts we get

$$\langle A\mathbf{u}, \boldsymbol{\phi}_K \rangle_{\mathrm{L}^2(K)} = \langle \operatorname{div} \mathbf{F}(\mathbf{u}), \boldsymbol{\phi}_K \rangle_{\mathrm{L}^2(K)} = - \langle \mathbf{F}(\mathbf{u}), \nabla \boldsymbol{\phi}_K \rangle_{\mathrm{L}^2(K)} + \sum_{F \in \mathcal{F}_K} \langle \mathbf{n}_{K,F} \cdot \mathbf{F}(\mathbf{u}), \boldsymbol{\phi}_K \rangle_{\mathrm{L}^2(F)}.$$

For $\mathbf{u}_\ell \in V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0}$ and $\boldsymbol{\phi}_\ell \in V^{\mathrm{dG}}_{K,\mathbf{p}\geq 0}$, we then define $\underline{A}_\ell \in \mathcal{L}(V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0}, V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0})$ cell-wisely, i.e.,

$$\left\langle \underline{A}_\ell \mathbf{u}_\ell, \boldsymbol{\phi}_{\ell,K} \right\rangle_{\mathrm{L}^2(K)} = - \left\langle \mathbf{F}(\mathbf{u}_{\ell,K}), \nabla \boldsymbol{\phi}_{\ell,K} \right\rangle_{\mathrm{L}^2(K)} + \sum_{F \in \mathcal{F}_K} \left\langle \mathbf{n}_{K,F} \cdot \mathbf{F}^*_{K,F}(\mathbf{u}_\ell), \boldsymbol{\phi}_{\ell,K} \right\rangle_{\mathrm{L}^2(F)}$$

where $\mathbf{n}_{K,F} \cdot \mathbf{F}^*_{K,F}(\mathbf{u}_\ell)$ is the upwind flux obtained from local solutions of Riemann problems (confer [141, section 2]). Using integration by parts again, we have

$$\left\langle \underline{A}_\ell \mathbf{u}_\ell, \boldsymbol{\phi}_{\ell,K} \right\rangle_{\mathrm{L}^2(K)} = \left\langle \operatorname{div} \mathbf{F}(\mathbf{u}_{\ell,K}), \boldsymbol{\phi}_{\ell,K} \right\rangle_{\mathrm{L}^2(K)} + \sum_{F \in \mathcal{F}_K} \langle \mathbf{n}_{K,F} \cdot \underbrace{\left(\mathbf{F}^*_{K,F}(\mathbf{u}_\ell) - \mathbf{F}(\mathbf{u}_{\ell,K}) \right)}_{=:\Delta \mathbf{F}_{K,F}(\mathbf{u}_\ell)}, \boldsymbol{\phi}_{\ell,K} \rangle_{\mathrm{L}^2(F)} \tag{2.14}$$

For the inner faces $F = \partial K \cap \partial K'$ it is a consistency requirement that the term in the sum over the faces $\mathbf{n}_{K,F} \cdot \Delta \mathbf{F}_{K,F}(\mathbf{u}_\ell)$ only depends on the jump $[\![\mathbf{u}_{\ell,K}]\!]_F = \mathbf{u}_{\ell,K'} - \mathbf{u}_{\ell,K}$. This is simply the case since for $\mathbf{u} \in \mathcal{D}(A)$ it is $\mathbf{n}_{K,F} \cdot \Delta \mathbf{F}_{K,F}(\mathbf{u}) = 0$ on all faces. Therefore, we see also for $\mathbf{u} \in \mathcal{D}(A) \cap \mathrm{H}^1(D, \mathbb{R}^J)$, $\mathbf{u}_\ell \in V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0}$ and $\boldsymbol{\phi}_\ell \in V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0}$ that

$$\langle A\mathbf{u}, \boldsymbol{\phi}_\ell \rangle_{\mathrm{L}^2(D)} - \langle \underline{A}_\ell \mathbf{u}, \boldsymbol{\phi}_\ell \rangle_{\mathrm{L}^2(D)} = 0 \quad \text{and} \quad \sum_{K \in \mathcal{K}} \left\langle \mathbf{n}_{K,F} \cdot \mathbf{F}^*_{K,F}(\mathbf{u}_\ell), \mathbf{u} \right\rangle_{\mathrm{L}^2(\partial K)} = 0.$$

Another consequence of the upwind flux is that it guarantees that the discrete operator is non-negative, i.e., $\langle \underline{A}_\ell \boldsymbol{\phi}_\ell, \boldsymbol{\phi}_\ell \rangle_{\mathrm{L}^2(D)} \geq 0$ for all $\boldsymbol{\phi}_\ell \in V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0}$. Having all that, we can conclude the semi-discrete convergence in the following theorem.

Theorem 2.31. *Suppose* $\mathbf{u} \in C^1((0,T), \mathcal{D}(A))$ *is a solution to* (2.11) *and* $\mathbf{u}_\ell \in C^1((0,T), V^{dG}_{\ell,\mathbf{p}\geq 0})$ *is a solution* *to* (2.13)*, and further, that the solution has the regularity* $\mathbf{u} \in \mathrm{L}^2((0,T), \mathrm{H}^s(D,\mathbb{R}^J))$ *with some* $1 \leq s \leq \mathbf{p}+1$ *as well as that there exists a constant* $C_{\underline{A}} > 0$*, such that*

$$\sum_{K\in\mathcal{K}} \sum_{F\in\mathcal{F}_K} \|\mathbf{n}_{K,F} \cdot \Delta\mathbf{F}_{K,F}(\boldsymbol{\phi}_\ell)\|^2_{\mathrm{L}^2(F)} \leq C_{\underline{A}} \left\langle \underline{A}_\ell \boldsymbol{\phi}_\ell, \boldsymbol{\phi}_\ell \right\rangle_{\mathrm{L}^2(D)}, \quad \forall \boldsymbol{\phi}_\ell \in V^{dG}_{\ell,\mathbf{p}\geq 0}.$$

Then, the following a priori semi-discrete error estimate holds

$$\|\mathbf{u}-\mathbf{u}_\ell\|^2_{\mathrm{L}^2([0,T],V)} \lesssim T C_{\underline{A}} h^{2s-1} \|\mathbf{u}\|^2_{\mathrm{L}^2([0,T],\mathrm{H}^s(D,\mathbb{R}^J))} + T\,\|\mathbf{u}(0)-\mathbf{u}_\ell(0)\|^2_V. \tag{2.15}$$

Proof. The proof can be found in [141, theorem 2]. □

2.3.3 Hyperbolic Contaminant Transport

The goal of this paragraph is to explain the discretization of the deterministic version of model problem 1.2 by neglecting the dependency on $\omega \in \Omega$. To this end, we realize that the transport model is a scalar ($J=1$) hyperbolic conservation law for a given vector field $\mathbf{q} \in \mathrm{W}^{1,\infty}(D,\mathbb{R}^{\mathbf{d}})$ with $\operatorname{div}\mathbf{q} = 0$. This can be seen by considering (2.11) with $\mathbf{F}(u) \coloneqq u(\mathbf{x},t)\mathbf{q}(\mathbf{x})$, i.e., $A\mathbf{u} = \operatorname{div}(\mathbf{q}u)$ with the operator A on the domain $\mathcal{D}(A) = \left\{u \in \mathrm{H}^1(D) \colon u = 0 \text{ on } \Gamma_{\mathrm{in}}\right\}$ mapping to $\mathrm{L}^2(D)$. Furthermore, we consider $f(\mathbf{x},t) \equiv 0$ for the rest of the paragraph but allow for an inflow on Γ_{in}. Thereby, we are in the setting we have been exploring in [28] and search for $u \in C^1((0,T),V)$, such that

$$\partial_t u + \operatorname{div}(\mathbf{q}u) = 0 \quad \text{on } D\times(0,T), \qquad u = u_{\mathrm{in}} \quad \text{on } \Gamma_{\mathrm{in}}\times(0,T), \qquad u(0) = u_0 \quad \text{on } D \tag{2.16}$$

describing the transport of a contaminating particle density in groundwater. We see by integrating over the time interval $[0,T]$ and the domain D that this is indeed a conservation law for the mass

$$\int_D u(\mathbf{x},T)\,\mathrm{d}\mathbf{x} = \int_D u_0(\mathbf{x})\,\mathrm{d}\mathbf{x} - \int_{\Gamma_{\mathrm{in}}}\int_0^T u_{\mathrm{in}}(t)\mathbf{q}\cdot\mathbf{n}\,\mathrm{d}t\mathrm{d}a - \int_{\Gamma_{\mathrm{out}}}\int_0^T \mathbf{u}(t)\mathbf{q}\cdot\mathbf{n}\,\mathrm{d}t\mathrm{d}a,$$

where $\Gamma_{\mathrm{out}} = \{\mathbf{x}\in\partial D \colon \mathbf{q}\cdot\mathbf{n} > 0\}$ and the energy

$$\int_D |u(\mathbf{x},T)|^2\,\mathrm{d}\mathbf{x} = \int_D |u_0(\mathbf{x})|^2\,\mathrm{d}\mathbf{x} + \int_{\Gamma_{\mathrm{in}}}\int_0^T |u_{\mathrm{in}}(\mathbf{x},t)|^2\,|\mathbf{q}\cdot\mathbf{n}|\,\mathrm{d}t\mathrm{d}a - \int_{\Gamma_{\mathrm{out}}}\int_0^T |\mathbf{u}(t)|^2\,|\mathbf{q}\cdot\mathbf{n}|\,\mathrm{d}t\mathrm{d}a.$$

Further, multiplying with a sufficiently smooth test function ϕ and integrating over the domain gives

$$\underbrace{\int_D \partial_t u\,\phi\,\mathrm{d}\mathbf{x}}_{=:\langle M\partial_t u,\phi\rangle_{\mathrm{L}^2(D)}} + \underbrace{\int_D \operatorname{div}(\mathbf{F}(u))\phi\,\mathrm{d}\mathbf{x} - \int_{\Gamma_{\mathrm{in}}} \Psi(u)\cdot\mathbf{n}\phi\,\mathrm{d}a}_{=:\langle Au,\phi\rangle_{\mathrm{L}^2(D)}} = \underbrace{-\int_{\Gamma_{\mathrm{in}}} \mathbf{F}(u_{\mathrm{in}})\cdot\mathbf{n}\,\phi\,\mathrm{d}a}_{=:\langle b,\phi\rangle_{\mathrm{L}^2(\Gamma_{\mathrm{in}})}}. \tag{2.17}$$

The goal is to present a dG discretization based on the above integral formulation and the abstract setting introduced in the previous paragraph 2.3.2. After this first step, a system of coupled ordinary differential equations (ODEs) has to be solved with time stepping methods which will be explained in section 2.4.

Model Problem 2.32. We search for an approximative solution $u_\ell \in V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0}$ using dG elements, such that

$$\langle \underline{M}_\ell \partial_t u_\ell, \phi_\ell \rangle_{\mathrm{L}^2(D)} + \langle \underline{A}_\ell u_\ell, \phi_\ell \rangle_{\mathrm{L}^2(D)} = \langle b_\ell, \phi_\ell \rangle_{\mathrm{L}^2(\Gamma_{\mathrm{in}})}$$

is satisfied. The discrete operators are in particular determined by

$$\langle \underline{M}_\ell u_\ell, \phi_\ell \rangle_{\mathrm{L}^2(D)} = \langle u_\ell, \phi_\ell \rangle_{\mathrm{L}^2(D)} \quad \text{and} \quad \langle b_\ell, \phi_\ell \rangle_{\mathrm{L}^2(\Gamma_{\mathrm{in}})} = - \langle \mathbf{n} \cdot \mathbf{F}(u_{\mathrm{in}}), \phi_\ell \rangle_{\mathrm{L}^2(\Gamma_{\mathrm{in}})}$$

and

$$\langle \underline{A}_\ell u_\ell, \phi_\ell \rangle_{\mathrm{L}^2(D)} = \langle \operatorname{div} \mathbf{F}(u_\ell), \phi_\ell \rangle_{\mathrm{L}^2(D)} + \sum_{F \subset \mathcal{F} \setminus \Gamma_{\mathrm{in}}} \langle \mathbf{n}_{K,F} \cdot \Delta \mathbf{F}_{K,F}(u_\ell), \phi_\ell \rangle_{\mathrm{L}^2(F)} + \langle \mathbf{n} \cdot \mathbf{F}(u_\ell), \phi_\ell \rangle_{\mathrm{L}^2(\Gamma_{\mathrm{in}})}$$

for $\phi_\ell \in V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0}$ and with $\Delta \mathbf{F}_{K,F}(\mathbf{u}_\ell) \coloneqq \mathbf{F}^*_{K,F}(\mathbf{u}_\ell) - \mathbf{F}(\mathbf{u}_{\ell,K})$. The numerical flux is given by

$$\mathbf{F}^*_{K,F}(u_\ell) = \begin{cases} \mathbf{F}(u_K) & \mathbf{q} \cdot \mathbf{n}_K > 0 \wedge F \in \mathcal{F} \setminus \partial D \\ \mathbf{F}(u_{K'}) & \mathbf{q} \cdot \mathbf{n}_K < 0 \wedge F \in \mathcal{F} \setminus \partial D \\ 0 & F \in \mathcal{F} \cap \partial D \setminus \Gamma_{\mathrm{in}} \end{cases},$$

which is obtained from local solutions of Riemann problems.

Proposition 2.33. *By construction of the numerical flux, the assumption of theorem 2.31 is satisfied. Hence, the semi-discrete solution satisfies the a priori error estimate* (2.15)*.*

Proof. The arguments can for example be found in [141, section 4.3]. □

Remark 2.34. a) In [81, section 5.6] and [70, chapter 2] a detailed analysis of the steady advection-reaction and in [70, chapter 3] the unsteady advection-reaction equation discretized with dG elements is presented. In particular, the equation

$$\partial_t u + \mathbf{q} \cdot \nabla u + \mu u = f$$

is considered with the assumptions

$$\operatorname{div} \mathbf{q}(\mathbf{x}) = 0, \quad \mathbf{q} \in \operatorname{Lip}(D, \mathbb{R}^{\mathbf{d}}) \quad \text{and} \quad \operatorname{div} \mathbf{q} \in \mathrm{L}^\infty(D)$$
$$\operatorname{Lip}(D) = \left\{ v \in C^0(D) \colon \left| v(\mathbf{x}_1) - v(\mathbf{x}_2) \right| \leq L_v \left| \mathbf{x}_1 - \mathbf{x}_2 \right| \right\}.$$

on flux field. Even though this theory is well-developed, we did not consider it in depth as the model is slightly different. However, by [44, section 1.3] we know if $\mathbf{q} \in \operatorname{Lip}(D, \mathbb{R}^{\mathbf{d}})$ holds, that $\mathbf{q} \in \mathrm{W}^{1,\infty}(D, \mathbb{R}^{\mathbf{d}})$ with $\|\nabla \mathbf{q}_d\| \leq L_{\mathbf{q}_d}$ for all $d \in \{1, \dots, \mathbf{d}\}$ under certain conditions on the domain. Henceforth, if we neglect the reaction part we are back in the initial setting.

b) It can be shown, e.g. in [134], that standard Lagrange finite elements applied to this model problem can not provide the necessary coercivity in an appropriate norm which leads to an unstable discretization method.

Remark 2.35. At this point, we discuss shortly what could be done to solve a parabolic transport-diffusion PDE as for example in [28]. We will omit the details on the particular procedure and only present two remarks on how to possibly do it.

a) To discretize a transport-diffusion equation we can combine the dG discretization with symmetric interior penalty of model 2.25 for the diffusion term, and the dG discretization with upwind flux for the transport term. This gives a stable discretization and is particularly interesting for a small diffusion term.

b) An alternative approach to discretize a parabolic transport-diffusion PDE with standard Lagrange finite elements is described in [148, Section 9.2]. However, for small diffusion terms this approach is not stable anymore (this is for example analyzed in [134, section 3]). To overcome this issue, in [47] the streamline diffusion method is proposed. Shortly speaking, a Petrov-Galerkin ansatz is chosen which introduces an artificial diffusion along the streamline. The advantage of this approach over the dG discretization is that the discrete system consists of less degrees of freedom while remaining stable.

2.3.4 The Acoustic Wave Equation

Next, we explain the discretization of the deterministic acoustic wave equation 1.3, i.e., for some initial value and boundary data, we consider the first order system

$$\begin{aligned} \rho\partial_t \mathbf{v} - \nabla p &= \mathbf{f} \\ \kappa^{-1}\partial_t p - \operatorname{div}\mathbf{v} &= g \end{aligned}$$

to find the velocity $\mathbf{v}$ and the pressure component p of the solution $\mathbf{u} = (\mathbf{v}, p)^\top$. We refer to [76, section 1.1.3-1.1.6] for a derivation of the acoustic wave equation out of elastic waves in isotropic media, as well as for the reformulation of the second order system into the first order system presented here. We further remark that this model can also be derived from compression waves in fluids or gases, but with another sign convention.

We can rewrite the acoustic wave equation in a compact operator form as done in [39, 141], i.e., depending on the material parameters ρ and κ, we define the operators

$$M := \begin{pmatrix} \rho & 0 \\ 0 & \kappa^{-1} \end{pmatrix} \Rightarrow M\mathbf{u} = \begin{pmatrix} \rho\,\mathbf{v} \\ \kappa^{-1}p \end{pmatrix}, \quad A := -\begin{pmatrix} 0 & \nabla \\ \operatorname{div} & 0 \end{pmatrix} \Rightarrow A\mathbf{u} = -\begin{pmatrix} \nabla p \\ \operatorname{div}\mathbf{v} \end{pmatrix}, \quad A_{\mathbf{n}}\mathbf{u} = \begin{pmatrix} -p\mathbf{n} \\ -\mathbf{n}\cdot\mathbf{v} \end{pmatrix} \tag{2.18}$$

We assume $\rho \in \mathrm{L}^\infty(D)$ with $\rho(\mathbf{x}) \geq \rho_0 > 0$ almost everywhere in D and $\kappa(\mathbf{x}) \equiv 1$. Henceforth, the system to model problem 1.3 can be formulated as linear hyperbolic first order system again which can be approximated with a Galerkin ansatz by multiplying with a test function $\boldsymbol{\phi} := (\boldsymbol{\varphi}, \psi) \in V$ and integrating over the domain D. Then, we can search for the solution in a finite dimensional dG space as in the following semi-discrete and deterministic model problem.

Model Problem 2.36. We search for an approximative solution $\mathbf{u}_\ell \in V_{\ell,\mathbf{p}\geq 0}^{\mathrm{dG}}$ using dG elements such that we satisfy

$$\langle \underline{M}_\ell \partial_t \mathbf{u}_\ell, \boldsymbol{\phi}_\ell \rangle_{\mathrm{L}^2(D)} + \langle \underline{A}_\ell \mathbf{u}_\ell, \boldsymbol{\phi}_\ell \rangle_{\mathrm{L}^2(D)} = \langle \mathbf{b}_\ell, \boldsymbol{\phi}_\ell \rangle_{\mathrm{L}^2(D)} - \langle \mathbf{g}_\ell, \boldsymbol{\phi}_\ell \rangle_{\mathrm{L}^2(\partial D)}$$

where the discrete operator $\underline{M}_\ell \in \mathcal{L}(V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0}, V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0})$ is a block diagonal positive definite matrix. We restrict ourselves to cell-wise constant material parameters, hence, we assume that we have cell-wise constant wave speeds $c_K = \sqrt{\kappa_K/\rho_K}$. The right-hand side $\mathbf{b}_\ell \in \mathrm{L}^2((0,T), V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0})$ and the matrix $\underline{M}_\ell$ are determined by Galerkin approximations, i.e., by

$$\langle \underline{M}_\ell \mathbf{u}_\ell, \boldsymbol{\phi}_\ell \rangle_{\mathrm{L}^2(D)} = \langle M \mathbf{u}_\ell, \boldsymbol{\phi}_\ell \rangle_{\mathrm{L}^2(D)}, \qquad \langle \mathbf{b}_\ell, \boldsymbol{\phi}_\ell \rangle_{\mathrm{L}^2(D)} = \langle \mathbf{b}, \boldsymbol{\phi}_\ell \rangle_{\mathrm{L}^2(D)}, \qquad \boldsymbol{\phi}_\ell \in V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0}.$$

Furthermore, for $\mathbf{u}_\ell = (\mathbf{v}_\ell, p_\ell)^\top \in V^{\mathrm{dG}}_{K,\mathbf{p}\geq 0}$ and $\boldsymbol{\phi}_\ell = (\boldsymbol{\varphi}_\ell, \psi_\ell)^\top \in V^{\mathrm{dG}}_{K,\mathbf{p}\geq 0}$, we define $\underline{A}_\ell = \sum_{K\in\mathcal{K}} \underline{A}_{\ell,K} \in \mathcal{L}(V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0}, V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0})$ with full upwind flux by

$$\begin{aligned}\big\langle \underline{A}_{\ell,K} \mathbf{u}_\ell, \boldsymbol{\phi}_\ell \big\rangle_{\mathrm{L}^2(K)} = &- \big\langle \nabla p_{\ell,K}, \boldsymbol{\varphi}_{\ell,K} \big\rangle_{\mathrm{L}^2(K)} - \langle \operatorname{div} \mathbf{v}_{\ell,K}, \psi_{\ell,K} \rangle_{\mathrm{L}^2(K)} \\ &- \sum_{F\in\mathcal{F}_K} \frac{1}{Z_K + Z_{K'}} \big\langle [\![p_{\ell,K}]\!]_F + Z_{K'} [\![\mathbf{v}_{\ell,K}]\!]_F \cdot \mathbf{n}_{K,F},\, \psi_{\ell,K} + Z_K \boldsymbol{\varphi}_{\ell,K} \cdot \mathbf{n}_{K,F} \big\rangle_{\mathrm{L}^2(F)}\end{aligned}$$

where $Z_K = \sqrt{\kappa_K \rho_K}$ is the impedance. Lastly, the boundary conditions $\mathbf{g} = (g_j)_{j=1}^J = (A_{\mathbf{n}} \mathbf{u})_{j=1}^J$, $g_j \in \mathrm{L}^2(\Gamma_j)$ are incorporated again with Galerkin approximations $\langle \mathbf{g}_\ell, \boldsymbol{\phi}_\ell \rangle_{\mathrm{L}^2(\partial D)} = \langle \mathbf{g}, \boldsymbol{\phi}_\ell \rangle_{\mathrm{L}^2(\partial D)}$.

Proposition 2.37. *By construction of the numerical flux, the assumption of theorem 2.31 is satisfied. Hence, the semi-discrete solution satisfies the a priori error estimate* (2.15).

Proof. Can be found in [141, section 4.3]. □

The resulting semi-discrete system can now be discretized in time, e.g. with the implicit mid-point rule, or be extended to a space-time setting.

Discretization in Time. We follow [39, section 3] and shortly outline the usage of the implicit mid-point rule with the time step size $\tau = T/N$ and the time steps $t_n = n\tau$, $n = 0, \ldots, N$, i.e., we construct a sequence of approximations $\mathbf{u}_\ell^n \in V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0}$ with the initial value of $\mathbf{u}_\ell^0 = \mathbf{u}_0$ by

$$\big(\underline{M}_\ell + \tfrac{\tau}{2} \underline{A}_\ell\big) \mathbf{u}_\ell^n = \big(\underline{M}_\ell - \tfrac{\tau}{2} \underline{A}_\ell\big) \mathbf{u}_\ell^{n-1} + \tau \mathbf{b}_\ell^{n-1/2} \quad \text{with} \quad \mathbf{b}_\ell^{n-1/2} := \mathbf{b}_\ell(n\tau - 1/2). \tag{2.19}$$

By [39, theorem 3.1] the above system is well-posed and therefore the implicit midpoint rule is applicable. We refer to section 2.4 for a short introduction to the class of implicit time stepping schemes.

Space-Time Discretization. We now describe the a full discretization on the space-time cylinder $Q := (0,T) \times D$ by using the ST-dG elements stated in section 2.2.5. The idea is to extend the above semi-discrete formulation with a dG discretization in time (we omit the details on this particular step, refer to [62, section 3] and remark that another sign convention for the wave equation was used in this particular source) which leads to a large algebraic system approximating the whole problem. As a start, we define the differential space-time operator $L \coloneqq M\partial_t + A$ where M and A are defined as previously. The *strong solution* is characterized with $\mathbf{b} \in \mathrm{L}^2(Q, \mathbb{R}^J)$, $\mathbf{u}_0 \in \mathrm{L}^2(Q, \mathbb{R}^J)$ and $\mathbf{g} = (g_j)_{j=1}^J$, $g_j \in \mathrm{L}^2((0,T) \times \Gamma_j, \mathbb{R}^J)$ by

$$L\mathbf{u} = \mathbf{b}, \qquad \mathbf{u}(0) = \mathbf{u}_0, \qquad (A_{\mathbf{n}}\mathbf{u})_j = g_j \quad \text{for} \quad j = 1, \dots J. \tag{2.20}$$

Since the adjoint satisfies $A^* = -A$, we also have $L^* = -L$ and therefore, we can look for the weak solution $\mathbf{u} \in \mathrm{L}^2(Q, \mathbb{R}^J)$ solving

$$\langle \mathbf{u}, L^*\boldsymbol{\phi} \rangle_{\mathrm{L}^2(Q)} = \langle \mathbf{b}, \mathbf{v} \rangle_{\mathrm{L}^2(Q)} + \langle M\mathbf{u}_0, \boldsymbol{\phi}(0) \rangle_{\mathrm{L}^2(D)} - \langle \mathbf{g}, \boldsymbol{\phi} \rangle_{\mathrm{L}^2((0,T)\times\partial D)}, \quad \forall \boldsymbol{\phi} \in V^* \subset C^1(\overline{Q}, \mathbb{R}^J)$$

We now develop the space-time discretization by considering the dG discretization in space from model 2.36 and the dG method in time of [62, section 3] giving a discrete adjoint space-time operator L_ℓ^*. We omit the details on the construction and only remark that the operator accounts for jumps at faces in space and time. Then, we search for $\mathbf{u}_\ell \in V_{\ell,\mathbf{p}_K,\mathbf{p}_I}^{\mathrm{ST-dG}}$ satisfying

$$\langle \mathbf{u}_\ell, L_\ell^*\boldsymbol{\phi}_\ell \rangle_{\mathrm{L}^2(Q)} = \langle \mathbf{b}_\ell, \boldsymbol{\phi}_\ell \rangle_{\mathrm{L}^2(Q)} + \langle \underline{M}_\ell\mathbf{u}_0, \boldsymbol{\phi}_\ell(0) \rangle_{\mathrm{L}^2(D)} - \langle \mathbf{g}, \boldsymbol{\phi}_\ell \rangle_{\mathrm{L}^2((0,T)\times\partial D)}, \quad \forall \boldsymbol{\phi}_\ell \in V_{\ell,\mathbf{p}_K,\mathbf{p}_I}^{\mathrm{ST-dG}}.$$

In [62, section 4] the well-posedness and the stability of the approach is analyzed and furthermore in [62, section 5] the convergence has been proven in the dG norm $\|\cdot\|_{\ell,\mathrm{dG}}$ (the definition can be found in [62, section 4] and [70, chapter 2 and 7]).

Theorem 2.38. *Assume that the strong solution to* (2.20) *is sufficiently smooth satisfying* $\mathbf{u} \in \mathrm{H}^s(Q, \mathbb{R}^J)$ *for some* $s \geq 1$. *Then, the error for the discrete solution* $\mathbf{u}_\ell \in V_{\ell,\mathbf{p}_K,\mathbf{p}_I}^{\mathrm{ST-dG}}$ *is bounded by*

$$\|\mathbf{u} - \mathbf{u}_\ell\|_{\ell,\mathrm{dG}} \lesssim h^{s-1/2} \|\mathrm{D}^s\mathbf{u}\|_{\mathrm{L}^2(Q)} + Th^{-1/2} \left\| \underline{M}_\ell^{-1/2}(\underline{M}_\ell - M)\partial_t\mathbf{u} \right\|_{\mathrm{L}^2(Q)},$$

where the hidden constant depends on the mesh regularity, the polynomial degree and the material parameters.

Proof. Can be found in [62, theorem 3]. □

Remark 2.39. In the main sources of this chapter, i.e., in [141, 76, 39, 62] the same approach is discussed for visco-acoustic, elastic and electro-magnetic waves. Hence, the presented discretization and the theory presented in the UQ chapter can be applied equally well for those equations. An indepth analysis and numerical experiments are left open for further work.

2.4 Numerical Solvers

In the previous section, we have explored how the deterministic PDEs of different kind are discretized. In order to find the coefficient vector $(\mu_1, \dots, \mu_N)^\top = \boldsymbol{\mu} \in \mathbb{R}^N$ representing the finite element solutions $\mathbf{u}_\ell = \sum_{n=1}^N \mu_n \boldsymbol{\psi}_n \in V_\ell$, we have to solve linear systems or systems of ordinary differential equations (ODEs). In this section, we discuss three things. First, we very briefly recall how preconditioned linear solvers and time stepping methods work. Following [28], we then will present a short discussion on the parallel linear algebra implemented in M++, and lastly we will embed this parallel algebra into the multi-mesh parallelization.

2.4.1 Linear Solvers and Time Stepping

Popular methods to find the solution $\boldsymbol{\mu} \in \mathbb{R}^N$ of sparse linear systems

$$\underline{A}\boldsymbol{\mu} = \mathbf{b} \quad \text{with} \quad \underline{A} \in \mathbb{R}^{N\times N} \quad \text{and} \quad \mathbf{b} \in \mathbb{R}^N$$

arising from FEMs are iterative Krylov subspace methods like the generalized minimal residual (GMRES) method or the conjugate gradient (CG) method. In particular in combination with an appropriate preconditioner $\underline{B} \approx \underline{A}^{-1}$ for the problem or multi-grid preconditioning, these methods can solve huge sparse systems up to an acceptable error in $\mathcal{O}(N)$ steps and are thereby the workhorse in high performance computing. Linear solvers and preconditioners are a central topic in numerical mathematics and even though the theory is well established, there is still a lot of research being done to increase performance and scalability for example by developing new multi-grid preconditioners. Advances in this field directly transmit to all other topics in numerical mathematics since linear systems have to be solved almost everywhere. We refer to standard literature like [131, 209] on linear solvers and to [152] for multi-grid methods and acknowledge that, even though this topic is mostly unconsidered in this thesis, it can be very impactful for our investigations.

Time stepping methods (or iterative time integration methods) are used to solve systems of ordinary differential equations of first order, i.e., problems where we search for $\mathbf{u}\colon [0,T) \to \mathbb{R}^N$ such that

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{u}(t) = \mathbf{F}(\mathbf{u}(t),t), \quad \mathbf{u}(0) = \mathbf{u}_0 \quad \text{with} \quad \mathbf{F}\colon \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}^N. \tag{2.21}$$

A popular family of methods to solve such systems are explicit and implicit *Runge-Kutta* methods which approximate solutions to (2.21) at distinct time points t_n with a sequence of $\mathbf{u}^n := \mathbf{u}(t_n)$ determined by

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \tau\sum_{i=1}^{S} b_i\mathbf{k}_i\,, \quad \text{where} \quad \mathbf{k}_i = \mathbf{F}(\mathbf{u}^n + \tau\sum_{j=1}^{S} a_{ij}k_j,\, t_n + c_i\tau) \quad \text{for} \quad i = 1,\dots,S. \tag{2.22}$$

We refer again to standard literature [131, section 76] and omit a detailed discussion on the topic. However, for *stiff* problems explicit methods ($a_{ij} = 0$ for $i \geq j$ in equation (2.22)) are only stable if the step size τ is sufficiently small, which motivates the usage of A-stable methods. These methods are implicit as all a_{ij} can be non-zero in (2.22). The downside of this approach is that the new iteration $\mathbf{u}^{n+1}$ is only *implicitly* given, i.e., this requires to solve a system of algebraic equations in each time step which can increase the cost of the method significantly. In the context of hyperbolic PDEs the stability criteria for explicit methods is expressed via the Courant–Friedrichs–Lewy (CFL) condition [64]

$$\frac{\tau}{h_\ell} \leq C_{\mathrm{CFL}}, \tag{2.23}$$

which, translated for wave equations, means that the time step must be smaller than the time the wave needs to travel from adjacent grid points in the mesh. Hence, if the distance between grid points h_ℓ is decreased, the time step size has to be decreased as well. We refer to the discussion in [141]. Lastly, we mention that the development and appropriate usage of time integration methods can be as impactful on the work of this thesis as improvements on the linear solvers. However, as this is not our main focus, we refer again to [141] for a more detailed discussion.

2.4.2 Parallel Linear Algebra

The coefficient vector $\boldsymbol{\mu} \in \underline{V}_\ell = \mathbb{R}^N$ representing the finite element solution $\mathbf{u}_\ell = \sum_{n=1}^N \mu_n \boldsymbol{\psi}_n \in V_\ell$ can be stored in parallel by the embedding $\underline{V}_\ell \to \underline{V}_{\ell,\mathcal{P}}$, where

$$\underline{V}_{\ell,\mathcal{P}} = \prod_{P \in \mathcal{P}} \underline{V}_{\ell,P} \quad \text{with} \quad \underline{V}_{\ell,P} = \mathbb{R}^{N_P} \quad \text{and} \quad N_P = |\{\mathbf{z} \in \mathcal{Z} \colon P \in \pi(\mathbf{z})\}|\,,$$

i.e., the space $\underline{V}_{\ell,\mathcal{P}}$ is overlapping at the points $\mathbf{z} \in \mathcal{Z}$, where $|\pi(\mathbf{z})| > 1$. Hence, we search for the parallel coefficient vector $\boldsymbol{\mu}_\mathcal{P} \in \underline{V}_{\ell,\mathcal{P}}$ (implemented in {13}) satisfying the parallelized system determined by $\underline{A}_\mathcal{P} = (\underline{A}_P)_{P\in\mathcal{P}}$ and $\mathbf{b}_\mathcal{P} = (\mathbf{b}_P)_{P\in\mathcal{P}}$ with

$$\underline{A}_P = \big(a_{\ell,P}(\boldsymbol{\psi}_i, \boldsymbol{\psi}_j)\big)_{i,j} \in \mathbb{R}^{N_P \times N_P} \quad \text{and} \quad \mathbf{b}_P = \big(f_{\ell,P}(\boldsymbol{\psi}_j)\big)_j \in \mathbb{R}^{N_P}.$$

Definition 2.40. a) A parallel vector $\boldsymbol{\nu}_\mathcal{P} = (\boldsymbol{\nu}_P)_{P\in\mathcal{P}} \in \underline{V}_\mathcal{P}$ is called *consistent*, if for all $P_1, P_2 \in \pi(\mathbf{z})$ the values coincide $\boldsymbol{\nu}_{P_1,\mathbf{z}} = \boldsymbol{\nu}_{P_2,\mathbf{z}}$.

b) A parallel vector $\boldsymbol{\nu}_\mathcal{P} = (\boldsymbol{\nu}_P)_{P\in\mathcal{P}}$ is called *additive*, if for all $P \in \pi(\mathbf{z})$ except for one the values are $\boldsymbol{\nu}_{P,\mathbf{z}} = 0$.

c) A parallel vector is called *accumulated*, if for all $P \in \pi(\mathbf{z})$ the values $\boldsymbol{\nu}_{P,\mathbf{z}}$ contain the sum of the true value over all processes.

Note that the inner products $(\underline{A}_\mathcal{P}\boldsymbol{\mu}_\mathcal{P}) \cdot \boldsymbol{\nu}_\mathcal{P}$ and $\mathbf{b}_\mathcal{P} \cdot \boldsymbol{\nu}_\mathcal{P}$ are well-defined for consistent vectors $\boldsymbol{\mu}_\mathcal{P}$ and $\boldsymbol{\nu}_\mathcal{P}$ despite the overlapping decomposition of $\underline{A}_\mathcal{P}$ and $\mathbf{b}_\mathcal{P}$. Furthermore, for iterative solvers, we construct preconditioners $\underline{B}_\mathcal{P} \colon \underline{V}^+_{\ell,\mathcal{P}} \to \underline{V}_{\ell,\mathcal{P}}$ which map additive to consistent vectors. This construction requires communication for which we refer to [28, 176, 177, 178] for details. This abstract concept extends to multi-level preconditioning, however, we omit the details for this as well and refer to [28, 75, 256].

2.4.3 Multi-sample Systems

The following only describes the case $M_\ell \leq |\mathcal{P}|$ of equation (2.4). However, everything applies without loss of generality for the case $M_\ell > |\mathcal{P}|$, since we then split in $i \in \mathbb{N}$ sequential computations without further interprocess communication, such that $\sum_i \widetilde{M}_{i,\ell} = M_\ell$ satisfying $\widetilde{M}_{i,\ell} \leq |\mathcal{P}|$.

By extending the finite element mesh on the input data $\{\mathbf{y}^{(m)}\}_{m=1}^{M_\ell}$ according to section 2.1, we also extended the definition of a finite element space in 2.10 and of the processor distribution mapping (2.5). Therefore, the parallel algebra building upon this has to be adapted as well, i.e., we search for a coefficient vector $(\mu_{1,1}, \dots, \mu_{1,N}, \dots, \mu_{M_\ell,1}, \dots, \mu_{M_\ell,N})^\top = \boldsymbol{\mu} \in \underline{V}_\ell = \mathbb{R}^{M_\ell \cdot N}$ representing the multi-sample finite element solution $(\mathbf{u}_\ell)_{m=1}^{M_\ell} = \left(\sum_{n=1}^N \mu_n^{(m)} \boldsymbol{\psi}_n^{(m)}\right)_{m=1}^{M_\ell} \in V_{\ell,\mathcal{P}}$. As in the previous paragraph this can be stored on a parallel computer with the embedding

$$\underline{V}_\ell \to \underline{V}_\mathcal{P} = \prod_{m=1}^{M_\ell} \prod_{P\in\mathcal{P}_k^{(m)}} \underline{V}_P^{(m)} \quad \text{with} \quad \underline{V}_P^{(m)} = \mathbb{R}^{N_P^{(m)}} \quad \text{and} \quad N_P^{(m)} = \left|\{\mathbf{z} \in \mathcal{Z}\colon P \in \pi(m,\mathbf{z})\}\right|.$$

Thereby, the whole computation can be expressed as an algebraic system on fully distributed memory, i.e., we search for $\boldsymbol{\mu} \in \underline{V}_\ell$, such that

$$\underbrace{\begin{pmatrix} \underline{A}_{\ell,\mathcal{P}_k}^{(1)} & 0 & \dots & 0 \\ 0 & \underline{A}_{\ell,\mathcal{P}_k}^{(2)} & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & \underline{A}_{\ell,\mathcal{P}_k}^{(M_\ell)} \end{pmatrix}}_{=\underline{A}_{\ell,\mathcal{P}}\in\mathbb{R}^{N_\ell\cdot M_\ell\times N_\ell\cdot M_\ell}} \cdot \begin{pmatrix} \boldsymbol{\mu}_{\ell,\mathcal{P}_k}^{(1)} \\ \boldsymbol{\mu}_{\ell,\mathcal{P}_k}^{(2)} \\ \vdots \\ \boldsymbol{\mu}_{\ell,\mathcal{P}_k}^{(M_\ell)} \end{pmatrix} = \underbrace{\begin{pmatrix} \mathbf{b}_{\ell,\mathcal{P}_k}^{(1)} \\ \mathbf{b}_{\ell,\mathcal{P}_k}^{(2)} \\ \vdots \\ \mathbf{b}_{\ell,\mathcal{P}_k}^{(M_\ell)} \end{pmatrix}}_{=\mathbf{b}_{\ell,\mathcal{P}}\in\mathbb{R}^{N_\ell\cdot M_\ell}}, \tag{2.24}$$

where each $\underline{A}_{\ell,\mathcal{P}_k}^{(m)} \in \mathbb{R}^{N_\ell\times N_\ell}$ is defined as

$$\underline{A}_{\ell,\mathcal{P}_k}^{(m)} = \left(\left(a_{\ell,P}^{(m)}(\boldsymbol{\psi}_i,\boldsymbol{\psi}_j)\right)_{i,j}\right)_{P\in\mathcal{P}_k^{(m)}} \quad \text{and} \quad \mathbf{b}_{\ell,\mathcal{P}_k}^{(m)} = \left(\left(f_{\ell,P}^{(m)}(\boldsymbol{\psi}_j)\right)_j\right)_{P\in\mathcal{P}_k^{(m)}}. \tag{2.25}$$

This results in the following method which we will later exploit as subroutine of the UQ algorithms.

Method 2.41. Given a PDE and suitable finite element discretization as well as the input data $\{\mathbf{y}_\ell\}_{\ell=0}^{M_\ell}$ parameterizing the PDE, the procedure to solve a multi-sample system (MSS) is given in the function below.

```
function MSS($M_\ell, \mathcal{P}$):
    $k \leftarrow$ use (2.4) with $(M_\ell, \mathcal{P})$
    $\underline{A}_{\ell,\mathcal{P}}, \mathbf{b}_{\ell,\mathcal{P}} \leftarrow$ for $m = 1,\dots,M_\ell$: assemble (2.25)
    $\boldsymbol{\mu}_{\ell,\mathcal{P}} \leftarrow$ solve $\underline{A}_{\ell,\mathcal{P}} \cdot \boldsymbol{\mu}_{\ell,\mathcal{P}} = \mathbf{b}_{\ell,\mathcal{P}}$
```

Remark 2.42. a) Since the parallelization is a function of M_ℓ and $|\mathcal{P}|$ minimizing the communication, it assembles the system (2.24), such that it minimizes the coupling, i.e., the system is decoupled for each sample and mildly coupled on the spatial domain. The system is sparse which is inherited from the sparsity of each finite element block.

b) The previously discussed linear solvers and time stepping methods build upon this parallelization. This highly distributed data structure enables us to employ implicit time stepping methods and space-time discretizations without running into memory issues.

c) As $\mathcal{P}$ (and ℓ) are fixed, the structure of the space $V_{\ell,\mathcal{P}}$ is only determined by the amount of samples M_ℓ. Thus, having a purely deterministic problem $M_\ell = 1$ (on level ℓ) results in a classical finite element matrix. Likewise, having a problem with highly variant input data, i.e., many samples are needed, the assembling of the matrix favours many small blocks, each representing a sample.

d) Since the matrix is decomposable, we can estimate the condition number with respect to the spectral norm of the whole system with

$$\mathrm{cond}_{\|\cdot\|_2}(\underline{A}_{\ell,\mathcal{P}}) = \left| \frac{\sigma_{\max}(\underline{A}_{\ell,\mathcal{P}})}{\sigma_{\min}(\underline{A}_{\ell,\mathcal{P}})} \right| \leq \max_{m=1,\dots,M_\ell} \left| \frac{\sigma_{\max}(\underline{A}^{(m)}_{\ell,\mathcal{P}_k})}{\sigma_{\min}(\underline{A}^{(m)}_{\ell,\mathcal{P}_k})} \right|,$$

where $\sigma_{\min}(\cdot)$ is the minimal and $\sigma_{\max}(\cdot)$ the maximal singular value of the matrix.

e) The parallel preconditioner $\underline{B}_{\mathcal{P}}$ potentially used in the last line of method 2.41 is constructed, such that it adapts to each sample block of the matrix only requiring communication on the subsets $\mathcal{P}_k^{(m)}$.

Lastly, we consider a multi-sample dG-system to present an example of the usage of this data structure for time stepping methods.

Example 2.43. We search for $(\mathbf{u}_\ell(t))_{m=1}^{M_\ell} \in V^{\mathrm{dG}}_{\ell,\mathcal{P}}$, such that

$$\underline{M}^{(m)}_\ell \partial_t \mathbf{u}^{(m)}_\ell(t) + \underline{A}^{(m)}_\ell \mathbf{u}^{(m)}_\ell(t) = \mathbf{b}^{(m)}_\ell(t), \quad t \in (0,T), \quad m = 1,\dots,M_\ell$$

where $V^{\mathrm{dG}}_{\ell,\mathcal{P}} = \prod_{m=1}^{M_\ell} V^{\mathrm{dG}}_{\ell,\mathcal{P}_k^{(m)}}$. Using (2.19) to discretize in time and [39, lemma 3.1], we can conclude that the above system is well-posed if we satisfy the assumptions on the input data of model 2.36 sample-wise. We refer to the upcoming section 3.3.4 for further details.

2.5 The FEM-Library M++

The open-source parallel finite element software M++ [28] has been developed within the last 15 years at the Karlsruhe Institute of Technology (KIT) by the group of Christian Wieners. As a research code for various applications like cardio-vascular simulations [94, 102], gas dynamic simulations for carbon capturing [150], nonlinear solid mechanics [29], dislocation dynamics [220, 254], computer assisted proofs [261] and full waveform inversion [39], it has grown a large array of features. This includes the parallelization via distributed point objects [257], parallel preconditioning [256, 177], space-time discretizations [75, 77] and a tutorial on scientific computing with examples of porous media model problems [28].

It has been a central piece in the doctoral thesis of Wolfgang Müller [192], Martin Sauter [216], Daniel Maurer [176], Stefan Findeisen [90], Jiping Xin [263], Andreas Schulz [219], Ekkachai Thawinan [248], Johannes Ernesti [83], Ramin Shirazi [225], Lydia Wagner [253], Daniel Ziegler [269], Julian Krämer [154], Jonathan Fröhlich [95] and Jonathan Wunderlich [261].

Motivated by all these applications and the rapid development in UQ, a main goal of this thesis was to contribute to this large software-project by equipping it with robust UQ algorithms. Surely, the models of the above applications can be enhanced with uncertainty one way or the other. A detailed technical description on the UQ extension of M++ will follow in section 3.6.

The development of the software is currently done by being involved in several research projects, namely on adaptive implicit space-time discretization for wave equations (CRC 1173 - project A3), dynamics of cardiac electrophysiological depolarization waves (CRC 1173 - project B7), seismic imaging by full waveform inversion (CRC 1173 - project C2), coupling methods for electro-mechanic models of the human heart (SPP 2311), nonlinear fracture dynamics (SPP 2256) as well as by faculty members of the mathematics department at KIT and external contributors to the open source code. Having a tool with such a diverse range of applications and such a variety of features does require careful software and algorithm engineering. Thereby, we discuss mainly two things in the remaining part of this section. Firstly, we describe the architecture of the code and the project structure in more detail and secondly, we explain how the project is maintained and how the development is pushed forward under *continuous delivery* (CD) workflows.

2.5.1 Software Architecture and Design

Throughout this document, we have distributed references to the implementation, which we want to give here once again to the main page of the git repository {14}. The `README.md` file gives a detailed installation guide of M++ on Linux systems equipped with Open MPI [96, 121], C++ 20 and CMake 3.5.1 or above. The software can also be build as docker-image.

By drawing the attention to figure 2.4, we explain the layered structure of the core library `Mpp` and the project organisation. We recall that the outline of this thesis is motivated by the code structure. Thereby, the reader of this document sees the theory behind FEM and UQ in the same order as the C++ compiler does.

`lib2_mesh`: After some preliminary definitions and essential setups, the second layer in `Mpp` mostly corresponds to section 2.1. This includes all objects of the finite element mesh like cells, faces, edges as well as the parallelization in spatial domain and across the data $\{\mathbf{y}^{(m)}\}_{m=1}^{M_\ell}$. Furthermore, the domain description {15} and the refinement rules of the mesh are implemented here as well.

`lib3_disc` and `lib4_fem`: The third and forth layer correspond to the sections 2.2 and 2.3, namely to the finite element spaces, including the shape functions {16} with the corresponding quadrature rules {17}, the dofs {18} and the coefficient vector {19} with the underlying parallelized data structure. Beyond the selection of finite elements we presented here, the software supports Argyris elements [44, page 76] implemented in {20}, curl elements [81, section 1.2.8] in {21}, Taylor-Hood elements [81, section 4.2.5] in {22} as well as further space-time {23}, mixed and vector-valued elements.

`lib5_solve`: The section 2.4 briefly discusses the application of linear solvers, preconditioners and time integrators. These methods are implemented in `lib5_solve` which currently offers parallel- (symmetric / block) Gauss-Seidel {24}, (damped / block) Jacobi {25}, incomplete *LU* decomposition {26} and multi-grid preconditioning {27}, iterative linear solvers like CG {28}, GMRES {29}, MINRES {30} and BiCGStab {31}. Furthermore, explicit and diagonal implicit Runge-Kutta methods are implemented in {32} alongside with exponential time integrators [140] in {33}. Lastly, we mention the implementation of eigen solvers in {34}.

`tests`: Testing, quality assurance and reliability are central concerns in software engineering. The `tests` library admits the same structure as the rest of the library, i.e., there is a test layer `test2_mesh` where most objects of `lib2_mesh` are tested in, a layer `test3_disc` which tests the objects in `lib3_disc` and so on. Test and behavior driven development (TDD/BDD) are known to be the fastest approaches to software

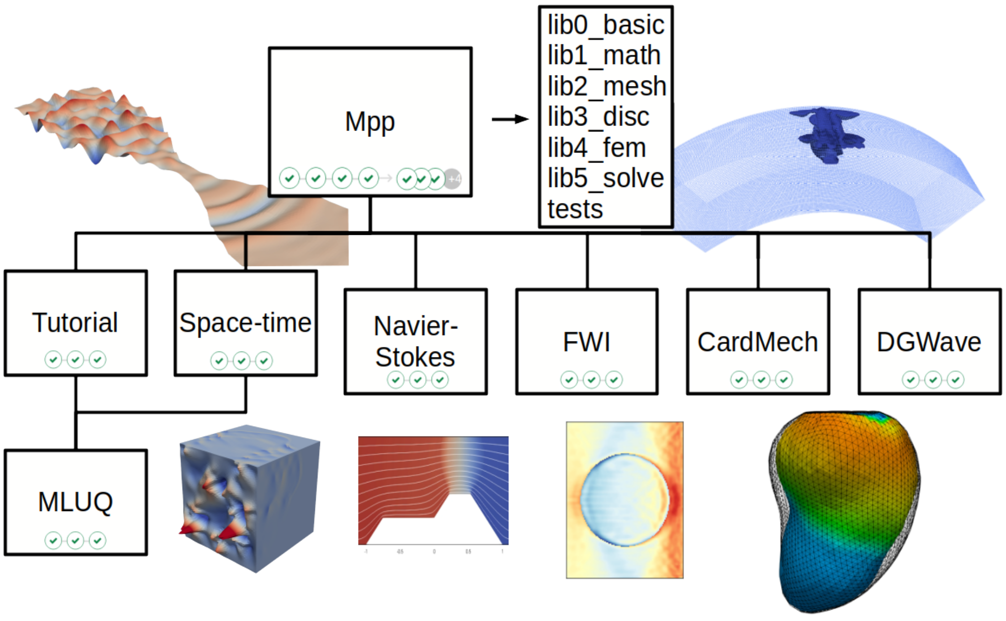


Figure 2.4: Project and library overview of M++.

development yielding the best quality of code [88]. Software solving pure mathematical problems are no exception to that, actually, in this domain it might have to be done even more rigorously.

`tutorial` and `spacetime`: The discrete and deterministic versions of the model problems from the introduction 1.3 and from section 2.3 are taken from the `tutorial` and `spacetime` libraries. We emphasize that the developed UQ methods can be applied easily to any other model implemented in M++.

`Navier-Stokes`, `FWI`, `CardMech`, `DGWave` and others: As mentioned in the introduction of this section, M++ has been part of many projects. Here, we only list the ones under current development. `Navier-Stokes` is a project concerned with the development of existence proofs for the Navier-Stokes equation using rigorous interval arithmetic, `FWI` provides methods for full waveform inversion in seismic imaging, in `CardMech` cardiovascular simulations are developed and executed, and in the project `DGWave` phase-field fracture resulting from elastic waves are investigated.

`MLUQ`: The whole chapter 3 and a detailed technical description in section 3.6 is dedicated to this part of the code.

Remark 2.44. Software architecture is traditionally associated with the fundamental structure and design choices of the system that are costly and time intense to change or to implement [26]. In some cases, this leads to much up front design and slow adaption to new requirements. Especially in research code development, the demands towards the software system might change rapidly from project to project and from PhD student to PhD student. Lengthy up front design and a slow pace of adaption are simply not suitable for research projects. By personal judgement, agile methods like *DevOps*, *Kanban* and *continuous delivery* (CD) (confer [88, 40, 205, 164] for the terminology) are the only way research software can be maintained and developed in a sustainable and long-lasting way. As suggested in the agile manifesto [30], by doing just enough architectural design up front and designing the system such that it should enable fast changes over anything, the system becomes a true asset to research or the research product in itself. The discussions in [6] and the software projects *Ginkgo* [7] and *OpenCarp* [15] have served as a great blue-print and inspiration to adapt our workflows towards software development under conscious *good scientific practice*. Concluded, the here presented structure of the project is nothing but a snap shot of M++, in fact, we encourage an ever evolving architecture, highly adaptive to the research requirement, making the above description on the libraries obsolete as soon as necessary.

We want to close this paragraph by giving an (incomplete) list of other open-source FEM software packages, each with its individual strengths and features. Many concepts such as the parallelized linear algebra, the interchangeability of finite element spaces, as well as the layered and modular architecture can all be found in some way or the other in these tools. They have served as a main inspiration for our own software development.

Example 2.45. a) The open source finite element library *deal.II* (Differential Equations Analysis Library) is a C++ library with similar applications as M++. Its most recent version is documented in [8]. It enjoys a long list of contributors (over 200 only on its GitHub main page) and applications.

b) *DUNE* (Distributed and Unified Numerics Environment) is an open source finite element library [27], following similar concepts as deal.II and M++. It is written to execute large scale finite element simulations on distributed memory.

c) The *FEniCSx* project enjoys much popularity because of its clean python front-end, mimicking weak formulations of the PDEs. An extensive description of the tool is given in [161].

d) *NGSolve* is another multi-physics finite element software. It provides a python front-end with similar ideas as FEniCSx. We refer to the documentation web site [197] for details.

e) We also mention *UG4*, developed at the Goethe Center for Scientific Computing at the University of Frankfurt [213]. The software *UG* has been a predecessor tool for *UG4*, *DUNE* and M++.

f) *PetsC* (Portable, Extensible Toolkit for Scientific Computation) provides a large array of preconditioners, linear solvers and time stepping methods. deal.II, DUNE, FEniCSx, OpenCarp and NGSolve all provide interfaces to PetsC. We refer to the user manual [20] and the webpage [19].

g) *Ginkgo* is a tool to solve sparse large-scale linear systems on a wide selection of hardware configurations, e.g. on GPU clusters [7]. In that sense, it is not a finite element library, but a solver library for finite element systems. Other tools like deal.II already provide interfaces to Ginko as subroutine to solve the arising linear systems. For us, their development of a *continuous delivery pipeline* including continuous benchmarking (CB) was a key motivation to enhance our development in that direction.

2.5.2 Continuous Delivery Workflow

Arguably the most important question arising in software development is: Does my software (change) solve the problem it was meant for? Say the task is to guide the user easier through an online store. How do we justify that a change of the system actually solved this imposed issue?

Luckily, the problems under consideration here are well-defined and provide lots of ways to quantify the improvements by a change. Especially since M++ is written to solve purely mathematical problems in HPC applications, we have to justify a change by answering these questions:

- Is it consistent with mathematical theory?
- Does it add a new application or method?
- Does it speed up the software?
- Does it fix flaws or errors?
- Does it improve the readability of the code?

At best, all questions are answered with yes when implementing a change. Most often though, these questions impose restrictions in a sense that previously achieved results have to be maintained. This means that a change has to lead to at least to the same speed of execution, at least the same readability of the code, at least the same number of executable applications and methods, and at least the same amount of consistent coverage of mathematical theory.

The *continuous delivery* (CD) workflow in software development is arguably the best way to justify each change to the system within the scope of the above questions. The idea is to keep the software always in a releasable state by automated testing, such that new features can be *continuously delivered* and the system can incrementally grow. Dave Farley's book [88] and the GitLab CI/CD documentation [111] have served as hands-on guides to realize a continuous delivery (CD) workflow and an associated CD pipeline with GitLab CI/CD. The CD pipeline is central to the workflow. It is a computer program running on a server which overtakes the automated testing and verification of the system, e.g. by executing the tests contained in the `tests` library. This goes beyond the verification of the core library by shipping it to other projects having dependencies on the library. Furthermore, the CD pipeline enables an automated deployment of the code on arbitrary hardware architectures and a (semi-)automated model building and validation by actually executing of the code on the hardware. We recall the goal of the method development from chapter 1 demanding robustness, repeatability and fast feedback, and emphasize that the CD pipeline is a key piece in achieving this goal. The CD pipeline will serve as our laboratory in the final investigations in chapter 4, allowing us to challenge a large array of hypotheses in a fully automated manner, and adjusting the method and the model incrementally to our findings.

In the following, we will explain the two tasks of verification and validation in a CD workflow (confer figure 2.5), leave a few remarks on the used hardware.

Verification. For the verification of the code we used techniques described in [151, 80, 215]. This includes checking the consistency of problems with exact solutions, verifying convergence results w.r.t. to certain discretization parameters, checking mathematical properties of a certain configuration, testing conserved quantities if they are actually conserved, testing for asymptotic behavior e.g. by Richardson extrapolation, testing for robustness and stability of a method by applying it to different problems, and continuously benchmarking

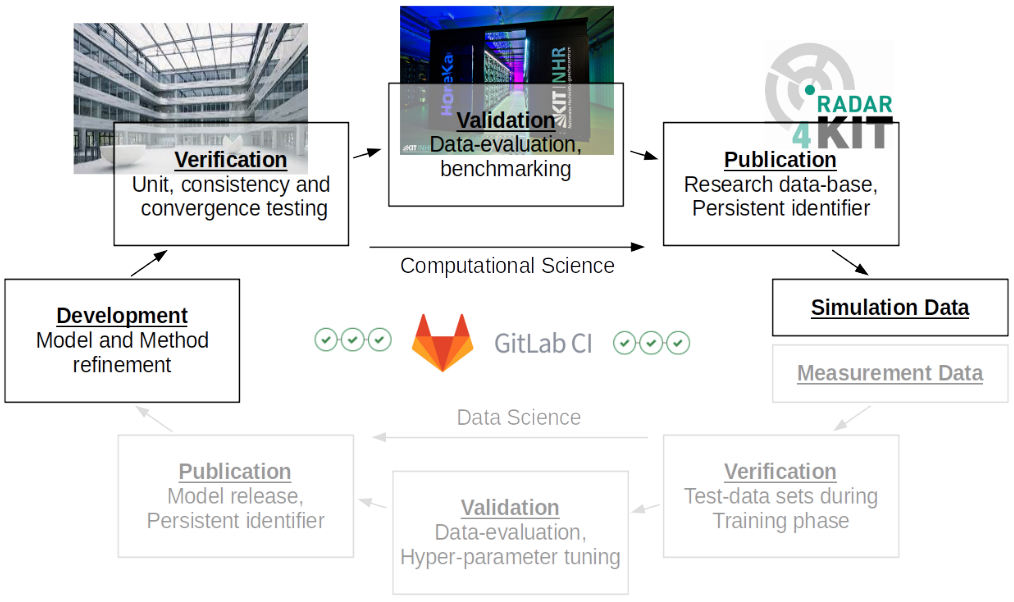

Verification
Unit, consistency and convergence testing
Validation
Data-evaluation, benchmarking
Publication
Research data-base, Persistent identifier
RADAR 4KIT
Computational Science
Development
Model and Method refinement
GitLab CI
Simulation Data
Measurement Data
Data Science
Publication
Model release, Persistent identifier
Validation
Data-evaluation, Hyper-parameter tuning
Verification
Test-data sets during Training phase

FIGURE 2.5: Semi-automated model-data cycle - workflow inspired by [6, 15].

the mathematically predicted performance of an algorithm. The verification task is henceforth important to demonstrate that the code solves the imposed mathematical problem as predicted by the theory. By translating properties of the mathematical system into tests on the behavior of the software system, verification is a task within the interface of computer science and mathematics [180]. Even better than verification of the code once is doing it *continuously* and throughout every stage of development in our CD workflow.

Validation. Model validation is essentially the question if the imposed mathematical model and the methods are appropriate for the system of interest [180]. A typical validation process involves the comparison of measured and simulated data. The better the match, the better is the representation by the computer model. We call our approach to the model validation semi-automated, because the match with experimental data and the following model refinement is missing. However, the CD pipeline allows us to generate fully automated large-scale simulation data from a broad selection of applications on demand, while the verification is executed as soon as the system changes. By doing so, we maximize the feedback on the development and hope to gain control over epistemic errors and improve incrementally by continuous (still hands-on) model refinement.

Publication. Subsequent to the verification and the validation is the publication task. The CD pipeline takes care of this last step by publishing verified and validated results accessing the *RADAR4KIT* service [212], creating a streamlined and well-defined process from development to research data and software publication with persistent identifiers, documented authorship and meta-data, as well as repeatable numerical results. This idea was taken from the OpenCarp [15] workflow and motivated by the *good scientific practice* guidelines described in [123].

Hardware. As pointed out in the introduction, we want a holistic method development including aspects from FEM, UQ, software engineering and hardware architectures. In that sense, we try to achieve a vertical integration of the hardware in the overall method development. After all, these are highly specialized algorithms requiring highly specialized hardware. We want to hide this complexity, such that whoever uses these methods should not be concerned about where and how the computation is actually executed. Already in 1968, a NATO software engineering conference took place. The complexity of computer programs at that time were the main matter of concern of this conference and the term *the software crisis* was coined. Edsger Dijkstra later described it by:

The major cause [of the software crisis] is that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem. In this sense the electronic industry has not solved a single problem, it has only created them, it has created the problem of using its products. -
Edsger W. Dijkstra, 1972

Since then, the problem as described by Dijkstra has grown exponentially by Moore's law. Fortunately, we have been granted access to three different computing clusters for which we have developed automated access, deployment and execution of the code. These computing clusters are:

PDE-Cluster: The PDE-Cluster is operated by the faculty for mathematics, KIT. It serves as the main machine for the code verification which includes the automated deployment of the library in the projects and the subsequent testing of them.

BwUniCluster 2.0: The Steinbuch Centre for Computing (SCC) at KIT operates the BwUniCluster 2.0. It is a massive parallel computer with a total of 848 nodes with a theoretical peak performance of 1.4 PetaFlops and a memory expansion of approximately 117 TB [49]. Nightly benchmarks are run on this machine.

HoReKa: The HoreKa (Hochleistungsrechner Karlsruhe) is operated by the NHR (Nationales Hochleistungszentrum) at SCC, KIT. Universities and research institutes from Germany can request access to this machine. It has a peak performance of 17 PetaFlops and a main memory of 220 TB [249].

By automating the workflow, we hide all the complexity associated with the hardware, the software and the mathematical difficulty to enable model building experts to focus on their concern. Thereby, we hide the *software crisis* associated with big computers and continuously deploy new methods to overcome these issues.

Clean Code. The term *clean code* (*cleanliness* of code) describes source code, documents, infrastructure setups and methods which are developed such that they are intuitively understandable and knowledge about the system can be acquired by simply reading the code, rather than an extensive documentation. It is characterized by being easy to read, easy to change and easy to extend. Common practices to achieve something as clean code are: keeping changes as small and as frequent as possible, versioning anything which is related to the software, automation and CD, using object names associated with the domain language (in

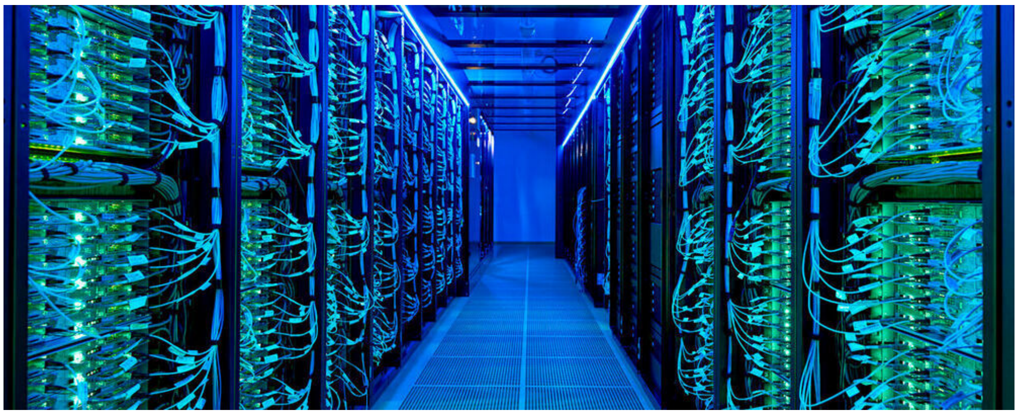


Figure 2.6: Hochleistungsrechner Karlsruhe (HoReKa) [249].

our case: mathematical terminology), practicing TDD and BDD (in our case: Implementing the verification of mathematical theory first), and sticking to popular software design principles.

However, *clean code* is a highly subjective and a non-quantifiable evaluation of the code (even though there are commercial and non-commercial tools like softwipe [267] which create a score for the code quality). Thereby, what is defined as *clean* and what *isn't* should be judged by all the people reading and using it.

3 Uncertainty Quantification

Uncertainty is everywhere. It enters every system as soon as measurements are involved, and it has to be appropriately taken into consideration. As Ryan McClarren describes in [180], *forward* UQ can be separated in several steps:

1. Identify a quantity of interest (QoI)
2. Identify the uncertain input data
3. Shrink the selection of uncertain inputs
4. Propagate the uncertainty through system
5. Determine how the uncertainty affects predictions

We want to structure this chapter in the same order as these steps indicate. In the introduction 1.3, we have determined what the QoIs within this thesis are, namely the expected value of some functional to a solution of a PDE. Thus, we want to start the chapter by further defining the uncertainty entering the system. In order to do so, we give a short review to stochastic processes, random fields and Bochner spaces in section 3.1. We remark that we take a probabilistic approach to UQ, but alternative tools like worst case analysis, interval arithmetic or fuzzy logic are employed for certain applications, too. Next, we will focus on the input data, introduce Karhunen-Loève (KL) expansions, sampling algorithms and sparse grids in section 3.2. Within the scope of this thesis, we will barely touch on the third step of UQ as described by Ryan McClarren. However, shrinking the dimension of the uncertain input data is crucial for efficient UQ and of great importance - it is just not the main matter of concern here. Section 3.3 will continue by stating the uncertainty entering the model problems 1.1-1.3 and outline their well-posedness with respect to these inputs. The remaining sections of this chapter are all concerned with the fifth step of UQ, discussing methods wich are capable to propagate uncertainty from input to output.

We further want to mention one more time the books on UQ by Sullivan [241] and Soize [229], but also the books in German by Fahrmeir, Künstler, Pigeot and Tutz [86] with a comprehensive overview on statistics and the book by León and Kiencke [165] on measuring techniques covering some sections of this chapter.

3.1 Random Fields and Bochner Spaces

In this section we introduce the definition of stochastic processes, random fields, Bochner spaces and a few essential properties. The theory is collected from [4, 218, 104] and [65]. For an overview on the probabilistic notation we refer to appendix A.2.

3.1.1 Introduction to Random Fields

Definition 3.1. a) Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, and let $D \subset \mathbb{R}^{\mathbf{d}}$ be a domain. A *random field* is a real valued (vector valued) mapping $Y\colon \Omega \times D \to \Xi \subset \mathbb{R}$ ($\mathbf{Y}\colon \Omega \times D \to \Xi \subset \mathbb{R}^J$) which, for every fixed $\mathbf{x} \in D$, is a measurable function of $\omega \in \Omega$. If $d = 1$, the random field is also called *stochastic process.*

b) For a fixed $\omega^{(m)} \in \Omega$ we call $\mathbf{y}(\omega^{(m)}, \cdot) \in \Xi$ a *realization* or a *sample*, which then is a deterministic function only depending on the location $\mathbf{x} \in D$. We will often denote a sample simply with $\mathbf{y}^{(m)}$.

Lemma 3.2. *For a collection of random fields $\{Y_m\colon \Omega \times D \to \mathbb{R}\}_{m=1}^M$ and a measurable function $\mathbf{f}\colon \mathbb{R}^M \to \mathbb{R}^J$, the component functions of $\mathbf{f}(Y_1(\omega, \mathbf{x}), \dots, Y_M(\omega, \mathbf{x}))$ are random fields, too. Thus, we can conclude that*

a) For $\{\alpha_m\}_{m=1}^M \subset \mathbb{R}$ the sum $\sum_{m=1}^M \alpha_m Y_m(\omega, \mathbf{x})$ is a random field.

b) The product $\prod_{m=1}^M Y_m(\omega, \mathbf{x})$ is a random field.

Proof. Confer [4, theorem 1.1 and corollary 1.1.1]. □

Definition 3.3. a) The *expectation field* of a random field $Y\colon \Omega \times D \to \Xi$ is given by

$$\mu_Y(\mathbf{x}) := \mathbb{E}[Y(\omega, \mathbf{x})] := \int_\Omega Y(\omega, \mathbf{x}) \mathrm{d}\mathbb{P}.$$

If $\mu_Y(\mathbf{x}) = 0$ for all $\mathbf{x} \in D$, the random field has *mean-zero.*

b) The *auto-covariance* of a random field $Y\colon \Omega \times D \to \Xi$ with $\mathbb{E}[(Y(\omega, \mathbf{x}))^2] < \infty$ for all $\mathbf{x} \in D$ gives the covariance of the process with itself at all pairs of points $\mathbf{x}_1, \mathbf{x}_2 \in D$

$$\mathrm{Cov}(\mathbf{x}_1, \mathbf{x}_2) = \mathbb{E}[(Y(\omega, \mathbf{x}_1) - \mu_Y(\mathbf{x}_1)) \cdot (Y(\omega, \mathbf{x}_2) - \mu_Y(\mathbf{x}_2))].$$

c) The covariance is closely related to the *variance field* $\sigma_Y^2(\mathbf{x}) = \mathrm{Cov}(\mathbf{x}, \mathbf{x})$.

d) The *auto-correlation* function $\rho(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathrm{Cov}(\mathbf{x}_1,\mathbf{x}_2)}{\sigma_Y(\mathbf{x}_1)\sigma_Y(\mathbf{x}_2)}$ is a normalization of the covariance.

Definition 3.4. For any $N \in \mathbb{N}$ and $n = 1, \dots, N$, let $\mathbf{x}_n \in D$ be arbitrary points in the domain and $c_n, \tilde{c}_n \in \mathbb{R}$. Then the covariance function on $D \times D$ is said to be *positive (semi-) definite* if

$$\sum_{n=1}^{N} \sum_{\tilde{n}=1}^{N} c_n c_{\tilde{n}} \,\mathrm{Cov}(\mathbf{x}_n, \mathbf{x}_{\tilde{n}}) \geq 0.$$

Theorem 3.5. *Auto-covariance functions of random fields are positive semi-definite.*

Proof. Confer [4, theorem 1.2] □

For some covariance functions, the positive semi-definiteness might be hard to verify. In this case, symmetry properties are helpful.

Definition 3.6. a) A random field $Y : \Omega \times D \to \Xi$ is called *strictly stationary* if for any choice of $\{\mathbf{x}_1, \dots, \mathbf{x}_K\}$, the finite-dimensional cumulative distribution function (CDF) is invariant under arbitrary translations $\tilde{\mathbf{x}}_k \in D$ with $\tilde{\mathbf{x}}_k + \mathbf{x}_k \in D$, i.e.,

$$F_{\mathbf{x}_1,\dots,\mathbf{x}_K}(y_1, \dots, y_K) = F_{\mathbf{x}_1+\tilde{\mathbf{x}}_1,\dots,\mathbf{x}_K+\tilde{\mathbf{x}}_K}(y_1, \dots, y_K).$$

b) A random field $Y : \Omega \times D \to \Xi$ is called *stationary (in a wide sense)* if its mean field is constant and its covariance function only depends on the separation vector $\tilde{\mathbf{x}} = \mathbf{x}_1 - \mathbf{x}_2$.

c) A stationary random field $Y : \Omega \times D \to \Xi$ is called *isotropic*, if the covariance function only depends on the norm of the separation vector $\|\tilde{\mathbf{x}}\| = \|\mathbf{x}_1 - \mathbf{x}_2\|$.

d) A stationary random field $Y : \Omega \times D \to \Xi$ is called *anisotropic* if the covariance function only depends on the norm $\|\tilde{\mathbf{x}}\|_{\underline{K}} = \sqrt{\tilde{\mathbf{x}}^\top \underline{K} \tilde{\mathbf{x}}}$ with $\underline{K}$ being positive semi-definite and not the identity.

The covariance function of isotropic and anisotropic random fields is positive definite. A strictly stationary random field is also stationary in a wide sense, the opposite is not necessarily true (cf. [4] for details). However, for Gaussian random fields the implication in the other direction is true, too.

3.1.2 Gaussian and Log-Normal Fields

The most prominent random fields are Gaussian random fields. They play an important role as their specification of the finite-dimensional distributions is very simple, they are a reasonable model for many natural phenomenas, e.g. as *white noise*, and the models are simply specified by the expectation field and the auto-covariance function.

Definition 3.7. A random field, where for any choice of $\{1, \dots, N\}$ the finite-dimensional CDF $F_{Y_1,\dots,Y_N}$ is that of a multivariate normal distribution, is called *Gaussian random field.* Thereby, for any choice of N and $\{\mathbf{x}_1, \dots, \mathbf{x}_N\} \in D$, we have

$$\mathbf{Y} = (Y(\omega, x_1), \dots, Y(\omega, x_N)) = (Y_1, \dots, Y_N)^\top \sim \mathcal{N}(\mu_{\mathbf{Y}}, \underline{C})$$

for some expectation field $\mu_Y(\mathbf{x})$ and some symmetric positive semi-definite covariance matrix $\underline{C} \in \mathbb{R}^{N \times N}$.

The existence of such a field is assured by Kolmogorov's extension theorem; the argument can be found in [4, Section 1.4]. The covariance function must be positive definite to ensure the existence of all finite-dimensional distributions. To sample from such a field, we introduce the following naive method.

Method 3.8. We want to sample from $\mathbf{Y} \sim \mathcal{N}(0, \underline{C})$ with an invertible covariance matrix. With the standard multivariate normal distribution $\mathbf{Z} \sim \mathcal{N}(0, \operatorname{diag}(1, \ldots, 1))$ and the decomposed covariance matrix $\underline{C} = \underline{R}\,\underline{R}^\top$ (e.g. with Cholesky), we can sample from the distribution by $\mathbf{Y} = \underline{R}\mathbf{Z}$.

Indeed, the vector $\mathbf{Y}$ has the required properties since $\mathbb{E}[\mathbf{Y}\mathbf{Y}^\top] = \mathbb{E}[\underline{R}\mathbf{Z}\mathbf{Z}^\top \underline{R}^\top] = \underline{R}\mathbb{E}[\mathbf{Z}\mathbf{Z}^\top]\underline{R}^\top = \underline{R}\operatorname{diag}(1 \ldots, 1)\underline{R}^\top = \underline{C}$ and $\mathbb{E}[\mathbf{Y}] = \mathbb{E}[\underline{R}\mathbf{Z}] = \underline{R}\mathbb{E}[\mathbf{Z}] = 0$. However, decomposing the covariance matrix stemming from a two-dimensional random field discretized e.g. on $\tilde{N} \times \tilde{N}$ grid points results in a dramatic computational effort. In particular, the Cholesky decomposition for the matrix of size $\tilde{N}^2 \times \tilde{N}^2$ results in the cost of $\mathcal{O}(\tilde{N}^6)$ which is beyond the scope of what is computable for problems of practical interest. Better methods will be introduced in section 3.2.

Definition 3.9. A continuous random variable $Y \colon \Omega \to \Xi \subset \mathbb{R}_{\geq 0}$ is called *log-normally* distributed, if $\log(Y) \sim \mathcal{N}(\mu_Y, \sigma_Y^2)$ is normally distributed. A random field $Y \colon \Omega \times D \to \Xi$ is called a *log-normal* field, if $\log(Y(\omega, \mathbf{x})) = g(\omega, \mathbf{x})$, where $g \colon \Omega \times D \to \mathbb{R}$ is a Gaussian random field.

The multiplicative mean of many independent, positive random variables is log-normally distributed. This is a result of the central limit theorem [86, section 7.1.2] in the log-domain. In the literature this is called Gilbert's law. A log-normal field satisfies $Y(\omega, \mathbf{x}) > 0$ almost surely in Ω and its values may vary over many orders of magnitude. This property is typical for the subsurface permeabilities and therefore log-normal fields will be used for the model problems 1.1-1.3.

Furthermore, we will assume that the Gaussian field is mean-zero, i.e., $\mathbb{E}[g(\omega, \mathbf{x})] = 0$ and that its covariance function can be represented as

$$\operatorname{Cov}(\mathbf{x}_1, \mathbf{x}_2) = \sigma^2 \exp\left(- \left\| \left(\frac{x_{1,d} - x_{2,d}}{\lambda_d} \right)_{d=1}^{\mathbf{d}} \right\|_2^\nu \right), \tag{3.1}$$

where σ^2 is the variance, $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_{\mathbf{d}})^\top$ is the correlation length in different spacial directions and ν can be used to smooth the covariance function. Since the function is not smooth if $\mathbf{x}_1$ approaches $\mathbf{x}_2$ for $\nu = 1$, realizations of the field $g(\omega, \mathbf{x})$ can be quite irregular. However, Kolmogorov's theorem (see for example [65]) states that all realizations of $g(\omega, \mathbf{x})$ are Hölder continuous for Hölder constants $t < \frac{1}{2}$ with respect to $\mathbf{x}$.

Lemma 3.10. *Let $g(\omega, \mathbf{x})$ be a Gaussian field with the covariance of (3.1). The samples of the log-normal field $\kappa(\omega, \mathbf{x}) = \exp(g(\omega, \mathbf{x}))$ belong to $C^t(\overline{D})$ almost surely for all $t < \frac{1}{2}$ and*

$$\|\kappa(\omega)\|_{C^t(\overline{D})} \leq (1 + 2\,|g(\omega)|_{C^t(\overline{D})})\kappa_{\max}(\omega).$$

Proof. This result can be found in [53]. □

Decreasing the correlation length increases the frequency of oscillations of $g(\omega, \mathbf{x})$, whereas increasing the variance increases the amplitude of those oscillations. This roughness in $g(\omega, \mathbf{x})$ is directly transmitted into $\kappa(\omega, \mathbf{x})$.

Remark 3.11. a) Smooth covariance functions, i.e., $\nu = 2$ lead to $g(\omega) \in C^1(\overline{D})$.

b) The results can even be extended to log-normal fields for which the underlying Gaussian field does not have mean-zero, under the assumption that this mean is sufficiently regular. Adding the mean $\mu(\mathbf{x})$ gives $\kappa(\omega, \mathbf{x}) = \exp(\mu(\mathbf{x}))\exp(g(\omega, \mathbf{x}))$ which still satisfies the assumptions as long as $\mu(\mathbf{x}) \in C^t(\overline{D})$ for t being the same as concluded from the smoothness of the field.

3.1.3 Bochner Spaces

For random variables which have values in Banach spaces, such as random field solutions to PDEs, we define Bochner spaces. To stay within the notation of the previous paragraph, we give the space of outcomes Ξ more structure by restricting it to Banach spaces. To this end, we use the following lemma which states that the norm of a random field is a random variable [65, proposition 1.2].

Lemma 3.12. *Let $(V, \|\cdot\|_V)$ be a separable Banach space and $Y\colon \Omega \times D \to V$ be a V-valued random field with a Borel σ-algebra on V. Then the mapping $\Omega \ni \omega \mapsto \|Y(\omega,\cdot)\|_V \in \mathbb{R}$ is a random variable.*

With that we can define Bochner spaces equipped with an appropriate norm since

$$\int_\Omega \|Y(\omega,\cdot)\|_V \,\mathrm{d}\mathbb{P} < \infty.$$

Bochner spaces are usually defined on an arbitrary measurable space, however, we will immediately identify this measurable space with a probability space leading to the following definition.

Definition 3.13. Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and $(V, \|\cdot\|_V)$ be a separable Banach space. We call

$$\mathrm{L}^q(\Omega, V) = \Big\{ Y\colon \Omega \to V\colon\ Y \text{ is strongly measurable with } \|Y\|_{\mathrm{L}^q(\Omega,V)} < \infty \Big\}$$

a *Bochner space* where norm is given by

$$\|Y\|_{\mathrm{L}^q(\Omega,V)} = \begin{cases} \left(\int_\Omega \|Y(\omega,\cdot)\|_V^q \,\mathrm{d}\mathbb{P} \right)^{1/q} & 1 \le q < \infty \\ \operatorname*{ess\,sup}_{\omega\in\Omega} \|Y(\omega,\cdot)\|_V & q = \infty \end{cases}.$$

In particular, we can also write

$$\|Y\|_{\mathrm{L}^1(\Omega,V)} = \mathbb{E}[\|Y\|_V] \quad \text{and} \quad \|Y\|_{\mathrm{L}^q(\Omega,V)} = (\mathbb{E}[\|Y\|_V^q])^{1/q}, \quad 1 \le q < \infty.$$

Remark 3.14. a) Bochner spaces are the right framework for solution and input spaces to the model problems 1.1-1.3. In particular, we will identify the Banach space V in definition 3.13 with Sobolev, Hölder or Lebesgue spaces.

b) If V is a Hilbert space, $\mathrm{L}^2(\Omega, V)$ is a Hilbert space with $\langle Y_1, Y_2 \rangle_{\mathrm{L}^2(\Omega,V)} = \int_\Omega \langle Y_1, Y_2 \rangle_V \, \mathrm{d}\mathbb{P}$.

c) Most of the properties of standard L^q spaces are transmitted on Bochner spaces. For example, the Cauchy-Schwarz inequality

$$\left| \langle Y_1, Y_2 \rangle_{\mathrm{L}^2(\Omega,V)} \right| \leq \|Y_1\|_{\mathrm{L}^2(\Omega,V)} \cdot \|Y_2\|_{\mathrm{L}^2(\Omega,V)}$$

and for $p, q \in [1, \infty]$ with $1/p + 1/q = 1$ the Hölder inequality are still valid, i.e.,

$$\|Y_1 Y_2\|_{\mathrm{L}^1(\Omega,V)} \leq \|Y_1\|_{\mathrm{L}^p(\Omega,V)} \cdot \|Y_2\|_{\mathrm{L}^q(\Omega,V)} \quad \Leftrightarrow \quad \mathbb{E}[\|Y_1 Y_2\|_V] \leq \left(\mathbb{E}[\|Y_1\|_V^p]\right)^{1/p} \cdot \left(\mathbb{E}[\|Y_2\|_V^q]\right)^{1/q}. \tag{3.2}$$

3.2 Uncertain Input Data

A crucial part of UQ is the modelling of the input data to the system of interest. As we will explain within this section, this can be done in various different ways and has a big impact on the computational demand of the overall method and the total error estimation. A task which is mostly unconsidered here in this thesis is the reduction of the dimensionality of the input data, which can reduce the computational demand significantly while keeping the error under control. However, with or without dimension reduction of the input, we need a way to create realizations $\mathbf{y}^{(m)}$ of the input data. In particular, we will briefly describe the generation of pseudo-random numbers on parallel processes, the representation of random fields by Karhunen-Loève (KL) expansions, the sampling of Gaussian random fields via circulant embedding, as well as a parametric representation of the probability spaces.

As a start, we have to ask the question how the uncertain input data is even characterized. Do we know the PDF or do we at least know basic statistics like the mean value or the covariance function? Where does the uncertainty enter the system, i.e., does it enter as an uncertain initial state, uncertain flux functions, uncertain material coefficients or boundary conditions and can it be modeled as a stochastic process or a random field? Whatever the nature of the uncertainty is, to be able to represent it in a computer we need the following crucial assumption.

Assumption 3.15. Throughout the thesis, we consider the *finite dimensional noise assumption* (FDNA), that is, the space of outcomes Ξ of any random variable $\mathbf{Y} \colon \Omega \to \Xi$ or any random field $\mathbf{Y} \colon \Omega \times D \to \Xi$ is of finite dimension K. Thereby, any sample can be represented by a vector $\mathbf{y}^{(m)} = (y_1, \dots, y_K)^\top \in \Xi \subset \mathbb{R}^K$.

This assumption is reasonable for most applications since many are actually only subject to finitely many random variables. However, even applications with an infinite-dimensional random input, a sufficient perturbation analysis can justify this assumption.

3.2.1 Parallel Pseudorandom Number Generation

Generating *true* random numbers is a non-trivial task. Although hardware random number generators, based on internal noisy physical states of the device, can be used to generate sequences of truly random numbers, pseudorandom number generators (PRNG) are widely in use due to their speed, their reproducibility and their independence of specialized hardware. PRNGs are algorithms which generate a sequence of numbers, which is purely determined by the initial state of the method named *seed*. Even though PRNGs are deterministic algorithms, a good PRNG produces a sequence such that no statistical test can distinguish between the output of the generator and a sequence of truly random numbers. However, when the sequence ends up in its seed again, it starts repeating itself. Thereby, PRNGs can approximate true randomness only up to a certain length of the sequence called the *period*. To make the matter worse, some PRNGs even produce shorter-than-expected periods for some seeds, while other seeds produce the predicted behavior. Especially for applications of PRNGs in cryptography, it must be impractical to try out all different seeds and to distinguish between an encrypted message and true randomness. However, well-designed PRNGs are also of key importance in Monte Carlo simulations since a repeating sequence is not indistinguishable from a sequence of iid samples, leading to biased solutions. Different fast and long-period PRNGs are discussed in [159]. Especially the ability to efficiently produce disjoint and uncorrelated streams of numbers with a single seed is essential for our purposes, since we also have to be sure that the parallel processes use uncorrelated streams of random numbers.

We use the software SPRNG5 [250] (scalable parallel random number generators library) which is statistically well tested and provides different algorithms. SPRNG5 is capable of producing different uncorrelated streams of random numbers and direct them on different processes while minimizing the communication in the MPI protocol. Furthermore, it assures reproducibility by a single global seed.

The parallel number generation is a crucial part to combine the integrated parallelism we explained in chapter 2 and the UQ methods introduced later in section 3.4 and 3.5. We refer to our wrapper implementation of SPRNG5 {35} from which we can sample pseudorandomly from a uniform distribution $Y \sim \mathcal{U}(0,1)$, and with that from arbitrary uniform distributions $\mathbf{Y} \sim \mathcal{U}(\mathbf{a},\mathbf{b})$ as well as arbitrary normal distributions $\mathbf{Y} \sim \mathcal{N}(\mu_{\mathbf{Y}}, \underline{C})$.

By being able to pseudorandomly sample from these basic probability distributions, we can construct more complex objects like stochastic processes and random fields. The following section explains this construction via Karhunen-Loève (KL) expansions.

3.2.2 Karhunen-Loève Expansion

The *Karhunen-Loève (KL) expansion* is a representation of a stochastic process or a random field $Y\colon \Omega \times D \to \mathbb{R}$ of second order as an infinite linear combination of orthogonal functions. Explanations of the KL expansions can for example be found in [241, section 11.1], [180, section 3.5] or [104, section 2.3]. To express the random field as KL expansion, we need to know the expectation field $\mu_Y(\mathbf{x})$, as well as the eigenvalues λ_k and the eigenfunctions ψ_k of the covariance operator $f \mapsto \int_D \operatorname{Cov}(\cdot, \mathbf{x}_2) f(\mathbf{x}_2) \mathrm{d}\mathbf{x}_2$. With uncorrelated random variables Z_k with mean-zero and unit variance, the random field can be expressed as a KL expansion by

$$Y(\omega, \mathbf{x}) = \mu_Y(\mathbf{x}) + \sum_{k=1}^{\infty} \sqrt{\lambda_k} \psi_k(\mathbf{x}) Z_k(\omega), \quad \omega \in \Omega, \quad \mathbf{x} \in D. \tag{3.3}$$

In particular, λ_k and ψ_k are eigenvalues and eigenfunctions satisfying

$$\int_D \mathrm{Cov}(\mathbf{x}_1, \mathbf{x}_2)\psi_k(\mathbf{x}_2)\mathrm{d}\mathbf{x}_2 = \lambda_k \psi_k(\mathbf{x}_1) \quad \text{with} \quad \sum_{k=0}^{\infty} \lambda_k^2 < \infty, \quad \lambda_1 \geq \lambda_2 \geq \cdots \geq 0, \tag{3.4}$$

where the eigenfunctions ψ_k span an orthonormal basis to $\mathrm{L}^2(D)$, i.e. $\langle \psi_k, \psi_{k'} \rangle_{\mathrm{L}^2(D)} = \delta_{kk'}$ for $k, k' = 1, \ldots, K$. Finding the eigenvalues and eigenfunctions in practice is a challenging task as it requires to find the spectrum of an integral operator. Nevertheless, the KL expansion is greatly useful in many applications, because it is a closed representation of a random field, a best approximation in $\mathrm{L}^2(\Omega)$ and $\mathrm{L}^2(D)$ and hence in some cases, it can be used to sample from the process. For example, if the random field is Gaussian and the eigenfunctions are known, it can be used to sample from the process with iid standard normally distributed random variables. We refer to [180, section 3.5] for a short explanation on the procedure in the non-Gaussian case.

In the following, we will focus without loss of generality on the mean-zero case $\mu_Y(\mathbf{x}) \equiv 0$ and investigate the existence of the expansion (3.3). To this end we restate Mercer's theorem.

Definition 3.16. A function $\mathrm{Ker}\colon D \times D \to \mathbb{R}$ is called *Mercer kernel*, if it is continuous, symmetric, i.e. $\mathrm{Ker}(\mathbf{x}_1, \mathbf{x}_2) = \mathrm{Ker}(\mathbf{x}_2, \mathbf{x}_1)$ for all $\mathbf{x}_1, \mathbf{x}_2 \in D$, and positive semi-definite, i.e., if for any choice of finitely many points $\mathbf{x}_1, \ldots, \mathbf{x}_N \in D$ the *Gram matrix* $\mathbf{G} := [\mathrm{Ker}(\mathbf{x}_n, \mathbf{x}_{n'})]_{n,n'=1}^{N}$ is positive semi-definite.

Theorem 3.17 (Mercer)**.** *Let* $\mathrm{Ker}\colon D \times D \to \mathbb{R}$ *be a Mercer kernel. Then there is an orthonormal basis* $\{\psi_k\}_{k\in\mathbb{N}}$ *of* $\mathrm{L}^2(D)$ *consisting of eigenfunctions of*

$$f \mapsto \int_D \mathrm{Ker}(\cdot, \mathbf{x}_2) f(\mathbf{x}_2)\mathrm{d}\mathbf{x}_2 \quad \textit{with non-negative eigenvalues} \quad \{\lambda_k\}_{k\in\mathbb{N}}.$$

The eigenfunctions corresponding to non-zero eigenvalues are continuous on D and the Mercer kernel has the representation

$$\mathrm{Ker}(\mathbf{x}_1, \mathbf{x}_2) = \sum_{k\in\mathbb{N}} \lambda_k \psi_k(\mathbf{x}_1)\psi_k(\mathbf{x}_2),$$

where the convergence of the sequence is absolute and uniform.

Proof. See [170, page 144]. □

With Mercer's theorem we can finally state the existence of the KL expansion.

Theorem 3.18 (Karhunen-Loève (KL) expansion)**.** *Let* $Y\colon \Omega \times D \to \Xi$ *be a square-integrable stochastic process or random field with mean-zero and a continuous and square-integrable covariance function. We have*

$$Y = \sum_{k\in\mathbb{N}} Z_k(\omega)\psi_k(\mathbf{x})$$

where $\{\psi_k\}_{k\in\mathbb{N}}$ *are orthonormal eigenfunctions of the covariance operator. The corresponding eigenvalues* $\{\lambda_k\}_{k\in\mathbb{N}}$ *are non-negative and the convergence of the series is in* $\mathrm{L}^2(\Omega)$*. Furthermore, the random variables*

Z_k are determined by

$$Z_k = \int_D Y(\mathbf{x})\psi_k(\mathbf{x})\,\mathrm{d}\mathbf{x} \quad \textit{and we have} \quad \mathbb{E}[Z_k] = 0, \quad \mathbb{E}[Z_k Z_{k'}] = \delta_{kk'}.$$

Proof. Can be found for example in [241, theorem 11.4]. □

Truncation of the Karhunen-Loève expansion. The truncated KL expansion reads as

$$Y(\omega, \mathbf{x}) = \mu_Y(\mathbf{x}) + \sum_{k=1}^{K} \sqrt{\lambda_k}\psi_k(\mathbf{x})Z_k(\omega), \quad \omega \in \Omega, \quad \mathbf{x} \in D. \tag{3.5}$$

Hence, the truncated is KL expansion is a surrogate model of the stochastic process satisfying the FDNA. This truncation introduces an error by itself. If the sequence of eigenvalues $\{\lambda_k\}_{k\in\mathbb{N}_0}$ does not decay fast enough as k grows, this truncation error becomes too large to be neglected in the total error analysis. An extended discussion on this is for example given in [118]. Lastly, we point out the similarity of the KL expansion to the singular value decomposition (SVD). As described in [180], the KL expansion can be interpreted as a SVD for a stochastic processes. In particular, we can find for both views the orthogonal decomposition and the data compression via truncation of the series. In many data-driven UQ applications, applying the SVD to find a representation of the input data in an orthogonal basis is often the first step to take which is similar to the first step in computational UQ, i.e., decomposing the stochastic process to get a closed representation of the input data.

3.2.3 Circulant Embedding

Even though the KL expansion is a very useful representation of a random field, in practice the eigenvalues and the eigenfunctions have to be known to create samples. Unfortunately, approximating them numerically can be a rather expensive task. Another approach to generate samples of a random field is the ciruclant embedding method. We refer to [79] for a short discussion on the usage of circulant embedding and KL expansions. Circulant embedding, firstly published in [72], is a highly efficient algorithm to generate iid samples from Gaussian random fields (confer 3.1). Roughly speaking, its basic idea is to exploit the special structure of embedded covariance matrices to perform a quick matrix decomposition by applying the fast Fourier transformation (FFT).

[The fast Fourier transform is] the most important numerical algorithm of our lifetime.
- Gilbert Strang, 1994, [235]

Example 3.19. As an introduction, suppose we want to sample from a stationary Gaussian random field on an unit torus $\mathcal{T}$, that is, the unit square $[0, 1] \times [0, 1]$ in which points on the opposite sides are identified with each other. This example was taken from [218, section 12] and gives a good intuition to the algorithm. We want to generate a stationary Gaussian random field on an equally spaced grid with $N \times N$ grid points on

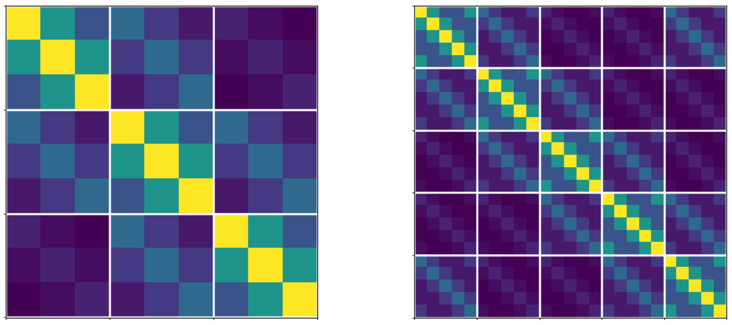


FIGURE 3.1: Illustration of a minimal odd circulant embedding (MOCE) of a BSTSTB matrix (left) for a random field on a 3×3 grid. The block structure in the BSTSTB matrix (left) as well as in the BCCB matrix (right) is indicated by the white lines. We remark the Toeplitz structure of each block on the left as well as the circulant structure of each block on the right.

the unit torus. Furthermore, we consider the covariance function

$$\mathrm{Cov}(\mathbf{x},\mathbf{y}) = \sigma^2 \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|_{\mathcal{T}}^{\nu}}{\lambda^{\nu}}\right) \quad \text{with} \quad \|\mathbf{x}\|_{\mathcal{T}} = \sqrt{\sum_{i=1}^{d} \min(|\mathbf{x}_i|, 1-|\mathbf{x}_i|)}.$$

The random field is in this case not just stationary but also isotropic since it is invariant under rotations. For our goal to create a stationary Gaussian random field in a $N \times N$ matrix, we have to create the $N^2 \times N^2$ covariance matrix $\underline{C}$ by plugging the grid points into the covariance function. This gives a block circulant matrix[1] with circulant blocks (BCCB) (confer the right image of figure 3.1 for an example). Thereby, the matrix is purely determined by the first row of blocks, where furthermore, each block is purely determined by its first row. We store this in the matrix $\underline{B}^{\text{rows}}$. This matrix can be decomposed with the Kronecker product of Fourier matrices $\underline{W}_N \in \mathbb{C}^{N\times N}$ with the entries $w_{m,n} = M^{-1/2}\exp(2\pi \mathrm{i}\, mn/M)$ for $m, n = 0, \dots, M-1$, i.e.

$$\underline{B}^{\text{rows}} = \underline{W}_{N,N}\underline{\Lambda}\underline{W}^{*}_{N,N} \quad \text{with} \quad \underline{W}_{N,N} := \underline{W}_N \otimes \underline{W}_N \quad \text{and} \quad \underline{\Lambda} = \mathrm{diag}(\lambda_1, \dots, \lambda_N),$$

λ_n being the eigenvalues of the covariance matrix for $n = 1, \dots, N$. These eigenvalues can be computed with $(\lambda_1, \dots, \lambda_N)^\top = N\underline{W}_{N,N}\underline{B}^{\text{rows}}$, where here $\underline{B}^{\text{rows}}$ is a $N \times N$ matrix containing the first row of each block. As demonstrated in method 3.8, we can sample from a Gaussian random field with the matrix $\mathbb{C}^{N\times N} \ni \underline{\mathbf{Z}} = \mathbf{Z}_1 + \mathrm{i}\mathbf{Z}_2$ with $\mathbf{Z}_1, \mathbf{Z}_2 \sim \mathcal{N}(0, \mathrm{diag}(1, \dots, 1))$ by

$$\underline{\mathbf{Y}} = \underline{W}_{N,N}\sqrt{\underline{\Lambda}}\underline{\mathbf{Z}}, \quad \text{where } \sqrt{\underline{\Lambda}} \text{ is computed component wise.}$$

Concluded, the sample $\underline{\mathbf{Y}}$ can be computed by applying the fast Fourier transform (FFT) on the product of the square root of the eigenvalues and $\underline{\mathbf{Z}}$. Then, the real part $\mathbf{Y}_1 = \mathrm{Re}(\underline{\mathbf{Y}})$ and the imaginary part $\mathbf{Y}_2 = \mathrm{Im}(\underline{\mathbf{Y}})$ are two independent real valued samples.

[1]A *circulant matrix* is a matrix where all rows contain the same entries. The entries are moved for each row one step to the right and wrapped around at the end of the matrix.

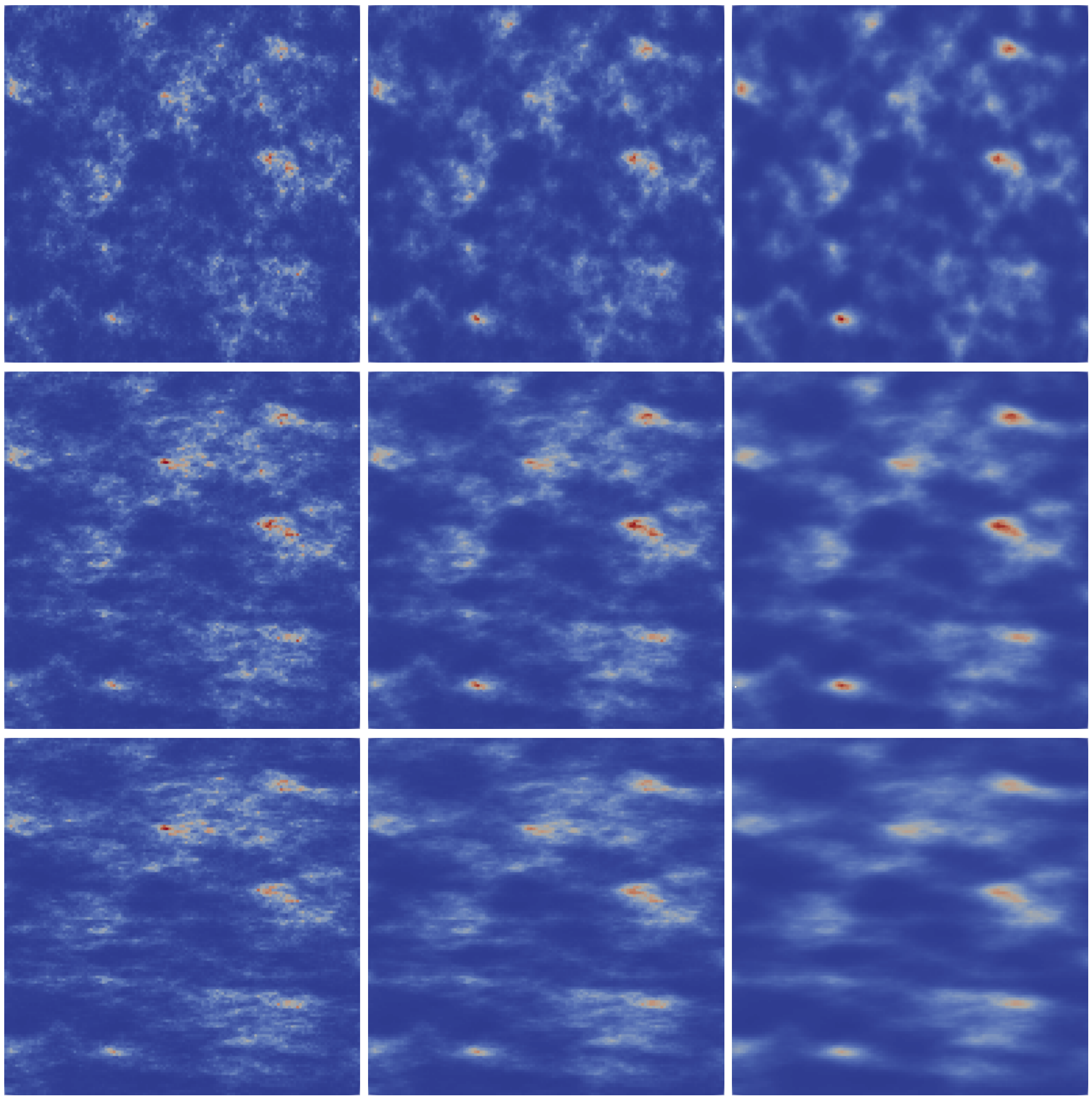


Figure 3.2: Log-normal fields with covariance function (3.6). Random fields are generated with circulant embedding on an equidistant grid with 128×128 cells and the same random seed. Images are smoothed with $\nu \in \{1.0, 1.4, 1.8\}$ from left to right and the correlation length is stretched in x_1 direction from top to bottom with $\lambda_1 \in \{0.05, 0.1, 0.15\}$.

Circulant embedding algorithm. As we have seen in example 3.19, the covariance matrix is easy to decompose, if the domain D is a unit torus. On a more general two-dimensional domain, the covariance matrix of stationary fields is not a BCCB matrix, but a block symmetric Toeplitz matrix[2] with (possibly symmetric) Toeplitz blocks (BST(S)TB). These matrices can be embedded in a BCCB matrix, such that we can decompose and proceed as in the example. We omit the details on the algebraic embedding of the

[2]A diagonal-constant matrix is also called *Toeplitz matrix*, i.e. the matrix is determined by a constant value for each diagonal. As a consequence it can be completely described by its first row and its first column.

matrices and refer to [244] and the example embedding of figure 3.1. There are two technical difficulties in this approach. First, the positive definitness of the embedded matrix is not guaranteed and second, the generated samples correspond to a larger domain. The second issue can be solved by extracting the upper left subdomain of the fields $\mathbf{Y}_1$ and $\mathbf{Y}_2$ with the initial domain size. In [72] it is mentioned that if the correlation between two points on the grid that are sufficiently far apart tends to zero, then a non-negative embedding exists. If this is not the case, the domain can be further extended by padding the BSTSTB matrix until the final embedded matrix is positive definite again. The final method can be summarized as follows.

Method 3.20. The circulant embedding method is done in two steps. First, the square roots of the eigenvalues have to be computed for each discretized domain represented by a mesh $\mathcal{M}_{\ell,\mathcal{P}}$ with $N \times N$ cells. This step only has to be done once. In a second step, arbitrarily many samples can be created for this field by using these eigenvalues.

```
function ComputeEV($\mathcal{M}_{\ell,\mathcal{P}}$):
    $\mathbb{R}^{N\times N} \ni \underline{A}^{\text{rows}}, \underline{A}^{\text{columns}} \leftarrow$ Cov($\mathcal{M}_{\ell,\mathcal{P}}$)
    $\mathbb{R}^{\widetilde{N}\times\widetilde{N}} \ni \underline{B}^{\text{rows}} \leftarrow$ MOCE($\underline{A}^{\text{rows}}, \underline{A}^{\text{columns}}$)
    $\mathbb{R}^{\widetilde{N}\times\widetilde{N}} \ni \underline{\Lambda}_\ell \leftarrow$ FFT($\underline{B}^{\text{rows}}$)
    if $\underline{\Lambda}_\ell > 0$:   return $\sqrt{\underline{\Lambda}_\ell}$
    else :        return ComputeEV($\widetilde{\mathcal{M}}_{\ell,\mathcal{P}}$)

function DrawSamples($\mathcal{M}_{\ell,\mathcal{P}}, \sqrt{\underline{\Lambda}_\ell}$):
    $\mathbb{C}^{\widetilde{N}\times\widetilde{N}} \ni \underline{\mathbf{Z}} \leftarrow$ SPRNG5($\mathcal{M}_{\ell,\mathcal{P}}$)
    $\mathbb{C}^{\widetilde{N}\times\widetilde{N}} \ni \underline{\mathbf{Y}} \leftarrow$ FFT($\sqrt{\underline{\Lambda}_\ell} \odot \underline{\mathbf{Z}}$)
    $\mathbb{R}^{N\times N} \ni \mathbf{Y}_1 \leftarrow \mathrm{Re}(\underline{\mathbf{Y}}[0:N, 0:N])$
    $\mathbb{R}^{N\times N} \ni \mathbf{Y}_2 \leftarrow \mathrm{Im}(\underline{\mathbf{Y}}[0:N, 0:N])$
    return $\mathbf{Y}_1 + \mu_{\mathbf{Y}}(\mathbf{x}),\ \mathbf{Y}_2 + \mu_{\mathbf{Y}}(\mathbf{x})$
```

Here, $\widetilde{\mathcal{M}}_{\ell,\mathcal{P}}$ is a mesh for an enlarged domain D. The implementation in C++ is given in {36}.

Remark 3.21. a) The operations $\sqrt{\cdot}$ and $\odot$ are component wise and `MOCE` is the minimal odd circulant embedding.

b) Only a small fraction of the covariance matrix has to be computed since it is completely determined by the first row $\underline{A}^{\text{rows}} \in \mathbb{R}^{N\times N}$ and the first column $\underline{A}^{\text{columns}} \in \mathbb{R}^{N\times N}$ of each block.

c) Likewise, we only have to store the first row of each block for the embedded BCCB matrix. The new matrix is of size $\widetilde{N} \times \widetilde{N}$ with $\widetilde{N} = (2N-1)$, if the MOCE was used.

d) In the two-dimensional case the step with the highest computational complexity is applying the FFT with $\mathcal{O}(\widetilde{N}^2 \log_2 \widetilde{N})$ (confer [218, section 12]).

e) We emphasize that this method is implemented on a multi-sample mesh $\mathcal{M}_{\ell,\mathcal{P}}$ using independent streams of pseudorandomly generated sequences of numbers on each MPI communicator.

f) The method works equally well for one or three dimensions. Further details can be found for example in [244, section 4.7].

g) The method is applicable to any covariance function as long as the covariance matrix can be embedded, such that its embedding is positive definite and circular. A selection of possible covariance functions is provided for example in [72] and [244, section 4.7].

h) Circulant embedding or similar methods are for example used in [55, 58, 118, 246, 185, 238] to generate log-normal fields.

i) The stochastic dimension in the algorithm is the same as the number of cells $|\mathcal{K}|$ in the mesh, however this is not the same as the stochastic dimension K of the KL expansion. In fact, the circulant embedding method usually uses way more random variables than the KL expansion.

To close this section, we present a few example realizations of a log-normal field in figure 3.2. We used the covariance function

$$\operatorname{Cov}(\mathbf{x}_1, \mathbf{x}_2) = \sigma^2 \exp\left(- \left\| \left(\frac{x_{1,d} - x_{2,d}}{\lambda_d} \right)_{d=1}^{\mathbf{d}} \right\|_2^{\nu} \right), \tag{3.6}$$

where $\boldsymbol{\lambda} = (\lambda_1, \lambda_2)^\top$ is the correlation length in x_1 and x_2 direction, σ is the process variance, ν is some smoothing parameter and the division is componentwise.

3.2.4 Sparse Grids

Sparse grids were initially designed by Sergei Smolyak [227] to integrate and interpolate high-dimensional functions. The underlying idea is to construct a sparse tensor product of a collection of one-dimensional quadrature (interpolation) points to be able to reduce the computational complexity arising with the approximation of high-dimensional functions. This is commonly known as the *curse of dimensionality*, coined by Richard Bellman in the context of *dynamic programming* problems [31] (we refer to 3.5 since Bellman's ideas on dynamic programming will play a key role in thesis). However, the usage of sparse grids in the stochastic collocation method (confer section 3.4) was developed to approximate high-dimensional integrals (confer [264, 12, 200, 201, 202, 13, 243, 190]).

To this end, we assume that the random variable $\mathbf{Y}\colon \Omega \to \Xi \subset \mathbb{R}^K$ satisfies the FDNA 3.15 and admits a PDF $f_{\mathbf{Y}}\colon \Xi \to [0, \infty)$. We then replace the probability space $(\Omega, \mathcal{F}, \mathbb{P})$ with a measure space $(\boldsymbol{\Gamma}, \mathcal{F}(\boldsymbol{\Gamma}), f_{\mathbf{Y}}(\mathbf{y})\mathrm{d}\mathbf{y})$, where $\mathcal{F}(\boldsymbol{\Gamma})$ denotes a σ-algebra of Borel sets on $\boldsymbol{\Gamma}$. We refer to the appendix A.2 for a definition of the used terminology. We will now follow [48, 234, 265] and briefly outline the idea of sparse grids and restrict ourselves to the canonical case $\boldsymbol{\Gamma} = [-1, 1]^K$. Using one-dimensional intervals $\Gamma_k = [-1, 1]$, we can use the tensor product to create $\boldsymbol{\Gamma} = \bigotimes_{k=1}^K \Gamma_k$. We now assume that we have discretized each one-dimensional domain Γ_k with $\{y^{(m)}\}_{m=1}^{M_k}$ points, hence, the total number of points in the tensorized grid is $M = \prod_{k=1}^K M_k$. For high dimensions K, the number of points grows rapidly and since these points are used for quadrature or interpolation rules, the cost of the approximation becomes too high very quickly. Therefore, instead of considering the full tensor product, we only take a small subset of nodes of the tensor product. This is illustrated in figure 3.3, where we used the software *TASMANIAN* [233] to generate the grids.

At this point, we want to omit a detailed discussion on the construction of the grids and rather refer to standard literature like [241, section 9.4] or [180, section 9.5]. However, figure 3.3 illustrates the dramatic reduction of points in the grid even though we have just been using a two-dimensional tensor product. We will pick up the topic of sparse grids again in section 3.4 outlining how these grids are used to provide collocation points for function interpolation and quadrature. We will denote the collection of K-dimensional collocation points constructed with sparse or tensor grids with $\{\mathbf{y}^{(m)}\}_{m=1}^M$.

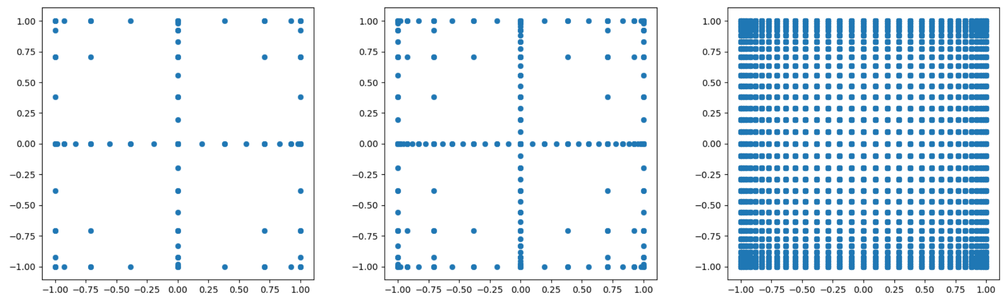


FIGURE 3.3: Sparse grids on refinement level four (left) and refinement level five (middle) as well as the full tensor grid on level five (right) constructed with Clenshaw-Curtis nodes. Further explanations can for example be found in [233].

3.3 Model Problems under Uncertainty

In this section, we present a sharper description of the model problems 1.1-1.3, i.e., we commit to the configuration of the input data. In particular, we denote where the uncertainty enters the system and what assumptions have to be satisfied by the trajectories in order to have well-posed problems. As done in all other sections, we start by explaining the elliptic case and end with the acoustic wave equation.

3.3.1 Elliptic Subsurface Diffusion Problem under Uncertainty

We start by considering the model problem 1.1. In particular, we will focus on a log-normally modeled permeability $\kappa \colon \Omega \times D \to \mathbb{R}$, which is considered to be an appropriate model [142] for the ground, and establish an existence and regularity theory as done in [55, 53, 246]. First, let us define for all $\omega \in \Omega$

$$\kappa_{\min}(\omega) \coloneqq \min_{\mathbf{x} \in \overline{D}} \kappa(\omega, \mathbf{x}) \quad \text{and} \quad \kappa_{\max}(\omega) \coloneqq \max_{\mathbf{x} \in \overline{D}} \kappa(\omega, \mathbf{x}).$$

Assumption 3.22. We make the following assumptions about the input data, similar to [246].

a) $\kappa_{\min} \geq 0$ almost surely and $\kappa_{\min}^{-1} \in \mathrm{L}^q(\Omega)$, for all $q \in (0, \infty)$.

b) $\kappa \in \mathrm{L}^q(\Omega, C^t(\overline{D}))$, for some $0 < t \leq 1$ and for all $q \in (0, \infty)$.

c) $f \in \mathrm{L}^{q_*}(\Omega, \mathrm{H}^{t-1}(D))$, $u_{\mathrm{D}} \in \mathrm{L}^{q_*}\left(\Omega, \mathrm{H}^{t+\frac{1}{2}}(\Gamma_{\mathrm{D}})\right)$ and $g_{\mathrm{N}} \in \mathrm{L}^{q_*}\left(\Omega, \mathrm{H}^{t-\frac{1}{2}}(\Gamma_{\mathrm{N}})\right)$, for $q_* \in (0, \infty]$.

For the sake of a lean representation, we will consider homogeneous Dirichlet and no Neumann boundary conditions from now on. However, the theory can be extended to non-homogenous and mixed boundary conditions under consideration of the above assumption and by standard techniques, e.g., as described in [125]. The Hölder continuity of the trajectories of the permeability in the second assumption assures that $\kappa_{\min}$ and $\kappa_{\max}$ are well-defined with $\kappa_{\max} \in \mathrm{L}^q(\Omega)$ and $0 < \kappa_{\min}$ as well as $\kappa_{\max} < \infty$ for almost all $\omega \in \Omega$. The second assumption can be weakened further to only piecewise Hölder continuous coefficients $\kappa(\omega, \cdot)$. The weaker assumptions are discussed in [246, section 5.2].

Model Problem 3.23. Consider 1.1 in a weak formulation for $\Gamma_{\mathrm{N}} = \emptyset$, $u_{\mathrm{D}} = 0$ on Γ_{D} and a fixed $\omega \in \Omega$, i.e., seek $u \in \mathrm{H}_0^1(D)$ such that

$$a_\omega(u(\omega,\cdot),\phi) = f_\omega(\phi), \quad \forall \phi \in \mathrm{H}_0^1(D),$$

where the parametrized bilinear form a_ω and the parametrized linear form f_ω are given by

$$a_\omega(u,\phi) = \int_D \kappa(\omega,\mathbf{x})\nabla u(\omega,\mathbf{x}) \cdot \nabla\phi(\mathbf{x})\,\mathrm{d}\mathbf{x} \quad \text{and} \quad f_\omega(\phi) = \int_D f(\omega,\mathbf{x})\phi(\mathbf{x})\,\mathrm{d}\mathbf{x}.$$

Since $\omega \in \Omega$ is fixed in model problem 3.23 but not uniformly bounded, everything stated from now on applies not uniformly but almost surely. Thus, $0 < \kappa_{\min}(\omega) \leq \kappa(\omega,x) \leq \kappa_{\max}(\omega) < \infty$ for all $x \in \overline{D}$ applies almost surely, too. The following lemma concludes ellipticity for our model problem and applies the lemma of Lax-Milgram (confer theorem A.5). Similar results can be found in [25, 53, 246].

Lemma 3.24. *The bilinear form of problem 3.23 is bounded and coercive with respect to the norm $|\cdot|_{\mathrm{H}^1(D)}$ for almost all $\omega \in \Omega$. Moreover, there exists a unique solution $u(\omega,\cdot) \in \mathrm{H}_0^1(D)$ and*

$$\|u(\omega,\cdot)\|_{\mathrm{H}^1(D)} \lesssim \frac{\|f\|_{\mathrm{H}^{t-1}(D)}}{\kappa_{\min}(\omega)}.$$

Proof. Let $\omega \in \Omega$ be fixed. The bilinear form is bounded and coercive by assumption 3.22, i.e.,

$$|a_\omega(u(\omega,\cdot),\phi)| \leq \kappa_{\max}(\omega)\,|u(\omega,\cdot)|_{\mathrm{H}^1(D)}\,|\phi|_{\mathrm{H}^1(D)} \quad \text{and} \quad a_\omega(u(\omega,\cdot),u(\omega,\cdot)) \geq \kappa_{\min}(\omega)\,|u(\omega,\cdot)|^2_{\mathrm{H}^1(D)}\,.$$

Furthermore, the linear form $f_\omega(\phi)$ is bounded by the Cauchy-Schwarz and the Poincaré inequality

$$\left|\int_D f(\omega,\mathbf{x})\phi(\mathbf{x})\,\mathrm{d}\mathbf{x}\right| \leq \|f(\omega,\cdot)\|_{\mathrm{H}^{t-1}(D)}\,\|\phi\|_{\mathrm{H}^{1-t}(D)} \lesssim \|f(\omega,\cdot)\|_{\mathrm{H}^{t-1}(D)}\,|\phi|_{\mathrm{H}^{1-t}(D)}\,.$$

Now we have by Lax-Milgram A.5 existence of a unique solution $u(\omega,\cdot) \in \mathrm{H}_0^1(D)$ for almost all $\omega \in \Omega$ to the variational model problem 3.23. From this and the coercivity, it can be concluded that

$$|u(\omega,\cdot)|_{\mathrm{H}^1(D)} \lesssim \frac{\|f\|_{\mathrm{H}^{t-1}(D)}}{\kappa_{\min}(\omega)}.$$

□

Theorem 3.25. *The weak solution to model problem 1.1 with assumption 3.22 and homogeneous boundary conditions is unique and belongs to $\mathrm{L}^q(\Omega,\mathrm{H}_0^1(D))$ for all $q < q_*$.*

Proof. Since $u(\omega,\cdot)$ is continuously depending on $\kappa(\omega,\cdot)$, the mapping $u\colon \Omega \to \mathrm{H}_0^1(D)$ is measurable. Now we use the result from lemma 3.24 and extend it on the space $\mathrm{L}^q(\Omega,\mathrm{H}_0^1(D))$ by using the assumptions 3.22 and Hölder's inequality (3.2) for $\frac{1}{q} = \frac{1}{q_*} + \frac{1}{p}$, with $p = \frac{q_* q}{q_* - q}$ being used for $\kappa_{\min}^{-1} \in \mathrm{L}^q(\Omega)$. □

It remains to show, that log-normally distributed random fields actually satisfy the assumption 3.22. We follow again [55, 53] and make use of lemma 3.10.

Corollary 3.26. *Let $g(\omega, \mathbf{x})$ be a Gaussian field with the covariance (3.6) and $\nu = 1$. Then the first two assumptions of 3.22 are satisfied for log-normal fields $\kappa(\omega, \mathbf{x}) = \exp(g(\omega, \mathbf{x}))$.*

Proof. First, $\kappa_{\min} \geq 0$ is fulfilled by definition. The proof of $\kappa_{\min}^{-1} \in \mathrm{L}^p(\Omega)$ can be found in [55] which is based on an application of Fernique's theorem (confer [65, section 2.2.1]). Furthermore, [55] also concludes that $g \in \mathrm{L}^p(\Omega, C^t(D))$ and $\kappa_{\max} \in \mathrm{L}^p(\Omega)$ for all $p \in (0, \infty)$ and $t < \frac{1}{2}$. With these results and lemma 3.10 the corollary above can be concluded. $\square$

Having existence under the assumptions 3.22 which are satisfied for log-normal fields, we can deduce the regularity results achieved in [55, 53, 246], which were also used for example in [101]. In section 4.1, we will combine these regularity results with finite element analysis of section 2.3. The regularity analysis tracks the influence of the random field $\kappa(\omega, \mathbf{x})$ and is thereby highly useful in the final experiments shown in section 4.1. We omit the details and restate the result.

Theorem 3.27. *Consider problem 3.23 on a polygonal domain D, assumptions 3.22 for some $0 < t \leq 1$ and homogeneous Dirichlet boundary conditions. Under further assumptions controlling corner singularities of the domain, it is*

$$\|u(\omega, \cdot)\|_{\mathrm{H}^{s+1}(D)} \lesssim \frac{\kappa_{\max}(\omega) \, \|\kappa(\omega, \cdot)\|_{C^t(D)}^2}{\kappa_{\min}(\omega)^4} \, \|f(\omega, \cdot)\|_{\mathrm{H}^{t-1}(D)},$$

for almost all $\omega \in \Omega$ and for all $0 < s < t$, such that s is also controlled by the corner singularities. Moreover, $u \in \mathrm{L}^q(\Omega, \mathrm{H}^{s+1}(D))$, for all $q < q_*$. If $t = 1$, then $u \in \mathrm{L}^p(\Omega, \mathrm{H}^2(D))$ and the above bound holds with $s = 1$.

Proof. The idea of the proof is to deduce everything for a fixed $\omega \in \Omega$ and then follow the regularity proof for elliptic PDEs as in [125, section 9.1]. The details on the procedure can be found in the appendix of [55, 53, 246]. $\square$

Since we have no access to true realizations of the random field and use circulant embedding, we recite the work done in [117, 101] where an interpolation result is given for trajectories of log-normal fields. This fits to the data generated by circulant embedding.

Lemma 3.28. *Given the assumptions 3.22, then for almost every $\omega \in \Omega$ it is*

$$\left\| \kappa(\omega, \mathbf{x}) - \Pi^{\mathrm{dG}}_{\mathbf{p} \in \{0,1\}} \kappa(\omega, \mathbf{x}) \right\|_{\mathrm{L}^\infty(D)} \lesssim h_\ell^t.$$

Proof. Can be found in [101, lemma 4.4]. $\square$

3.3.2 Hyperbolic Conservation Laws under Uncertainty

We now extend section 2.3.2 to random fields. This includes the definitions of conservation laws 2.28, hyperbolicity 2.29 and weak solutions 2.30. Similar work has been done for example in [239, 185], however, we stick to the notation of [141, 75, 39, 62, 76]. As a start, we define a conservation law under uncertainty and state assumptions on the input data. To this end, we introduce the flux operator under uncertainty acting on the random field solution by

$$\operatorname{div} \mathbf{F}(\omega, \mathbf{u}(\omega)) \coloneqq \sum_{d=1}^{\mathbf{d}} \underline{B}_d(\omega)(\partial_d \mathbf{u}(\omega)) = A(\omega)\mathbf{u}(\omega).$$

With that, we can define a conservation law under uncertainty.

Definition 3.29. We call a problem: seek a random field solution $\mathbf{u}\colon \Omega \ni \omega \mapsto \mathbf{u}(\omega, \mathbf{x}, t)$, i.e., a measurable mapping from $(\Omega, \mathcal{F})$ to $C^1([0,T], V)$ with $V \subset \mathrm{L}^2(D)$ such that

$$M(\omega)\partial_t \mathbf{u}(\omega) + \operatorname{div} \mathbf{F}(\omega, \mathbf{u}(\omega)) = \mathbf{b}(\omega) \quad \text{and} \quad \mathbf{u}(\omega, 0) = \mathbf{u}_0(\omega), \tag{3.7}$$

a *conservation law under uncertainty* if $\mathbf{b} \equiv \mathbf{0}$ or a *balance law under uncertainty* if $\mathbf{b} \not\equiv \mathbf{0}$.

Assumption 3.30. Very similar to [185, theorem 2.5] and [239, theorem 2] we assume:

a) the initial data satisfies $\mathbf{u}_0 \in \mathrm{L}^q(\Omega, \mathrm{L}^2(D, \mathbb{R}^J))$,

b) the right-hand side satisfies $\mathbf{b} \in \mathrm{L}^q(\Omega, \mathrm{L}^2(Q, \mathbb{R}^J))$,

c) the mass operator satisfies $M \in \mathrm{L}^q(\Omega, \mathrm{L}^\infty(D, \mathbb{R}^{J\times J}_{\mathrm{sym}}))$,

d) and lastly, we assume that $\underline{B}_1(\omega), \dots, \underline{B}_{\mathbf{d}}(\omega)$ are $\mathbb{R}^{J\times J}_{\mathrm{sym}}$ valued random fields, i.e.,

$$\underline{B}_d \in \mathrm{L}^q(\Omega, \mathrm{L}^\infty(D, \mathbb{R}^{J\times J}_{\mathrm{sym}})) \quad \text{for all} \quad d = 1, \dots, \mathbf{d}$$

and that all $\underline{B}_d$ are independent of $\mathbf{u}_0$, $\mathbf{b}$ and M on $(\Omega, \mathcal{F}, \mathbb{P})$.

Definition 3.31. A system of the form (3.7) is called *hyperbolic*, if it is almost surely hyperbolic according to definition 2.29, i.e., for almost every fixed $\omega \in \Omega$ the matrix $\underline{B}(\omega)$ of equation (2.12) has J real eigenvalues $\lambda_1(\mathbf{n}, \omega) \le \lambda_2(\mathbf{n}, \omega) \le \dots \le \lambda_J(\mathbf{n}, \omega)$ with a complete family of eigenvectors, i.e., the system is almost surely diagonalizable.

Definition 3.32. The random field solution $\mathbf{u}\colon \Omega \ni \omega \mapsto \mathbf{u}(\omega, \mathbf{x}, t)$, i.e., the measurable mapping from $(\Omega, \mathcal{F})$ to $\mathrm{L}^2((0,T) \times D, \mathbb{R}^J)$ is called *weak solution under uncertainty* to (3.7), if $\mathbf{u}(\omega, \cdot, \cdot)$ is almost surely a weak solution to (2.11). Thus, for almost every fixed $\omega \in \Omega$ the weak solution satisfies

$$\int_D M(\omega, \mathbf{x})\mathbf{u}_0(\omega, \mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{x}, 0)\, \mathrm{d}\mathbf{x} + \int_{(0,T)\times D} \big(M(\omega, \mathbf{x})\mathbf{u}(\omega, \mathbf{x}, t) \cdot \partial_t \boldsymbol{\phi}(\mathbf{x}, t) + \mathbf{F}(\omega, \mathbf{u}(\omega, \mathbf{x}, t)) \cdot \nabla \boldsymbol{\phi}(\mathbf{x}, t)\big)\, \mathrm{d}t\, \mathrm{d}\mathbf{x}$$
$$= \int_{(0,T)\times D} \mathbf{b}(\mathbf{x}, t)\boldsymbol{\phi}(\mathbf{x}, t) \mathrm{d}t\, \mathrm{d}\mathbf{x}$$

for all test functions with $\boldsymbol{\phi} \in \big\{\mathbf{v} \in C^1((0,T) \times D, \mathbb{R}^J)\colon \mathbf{v}|_{\partial D} = 0 \text{ and } \mathbf{v}(\cdot, T) = 0\big\}$.

Theorem 3.33. *Suppose assumptions 3.30 are satisfied and the balance law (3.7) is hyperbolic. Then, for $T < \infty$ equation (3.7) admits a random weak solution denoted with*

$$\mathbf{u}\colon \Omega \to \mathrm{L}^2((0,T) \times D, \mathbb{R}^J), \quad \omega \mapsto \mathbf{u}(\omega, \cdot, \cdot)$$

where $\mathbf{u}(\omega, \cdot, \cdot)$ is the solution to a deterministic realization of (3.7) which is for all $t \in [0,T]$ almost surely controlled by the input data

$$\|\mathbf{u}(\omega, \cdot, t)\|_{\mathrm{L}^2(D)} \lesssim \|\mathbf{u}_0(\omega, \cdot)\|_{\mathrm{L}^2(D)} + t\, \|\mathbf{b}(\omega, \cdot, t)\|_{\mathrm{L}^2(D)} ,$$

where the hidden constant depends on $\omega \in \Omega$ the flux and the mass operator.

Proof. The theorem was taken from [185, theorem 2.5]. The proof is outlined in [239, theorem 2] following the steps described in [183]. □

Semi-Discretization of Hyperbolic Conservation Laws Under Uncertainty. We will omit the detailed description of the semi-discrete formulation under uncertainty and note that we apply everything as in section 2.3.2 for a fixed $\omega \in \Omega$.

3.3.3 Hyperbolic Contaminant Transport under Uncertainty

Now we can consider model problem 1.2, where we focus on the particular case of a stochastically modelled and divergence free flux field $\mathbf{q}\colon \Omega \times D \to \mathbb{R}^{\mathbf{d}}$. A similar problem was considered in [156]. For the sake of a simple representation we also consider no source term, no inflow boundary condtion but allow for randomly distributed, differentiable initial conditions $u_0 \in \mathrm{L}^q(\Omega, C^1(D))$. Thereby, (2.16) becomes the following model problem.

Model Problem 3.34. For a fixed $\omega \in \Omega$, we consider the formulation based on (2.17), i.e.,

$$\langle M \partial_t u(\omega), \phi \rangle_{\mathrm{L}^2(D)} + \langle A(\omega) u(\omega), \phi \rangle_{\mathrm{L}^2(D)} = 0 \quad \text{for} \quad t \in (0,T] \quad \text{and} \quad u(\omega, 0) = u_0(\omega). \tag{3.8}$$

with $A(\omega)u(\omega) = \mathrm{div}(\mathbf{q}(\omega)u(\omega))$.

We are not tied to this particular configuration and emphasize that uncertain boundary conditions or right-hand sides are easily realizable under the setting considered in section 2.3.3 and section 3.3.2. The vector valued random fields $\mathbf{q}\colon \Omega \times D \to \mathbb{R}^{\mathbf{d}}$ are generated by solving the elliptic subsurface diffusion problem with log-normally distributed permeability such as described in section 3.3.1. In particular, it is solved with mixed finite elements (confer model 2.22 and model 2.23) to ensure that the flux is divergence free and $\mathbf{q}(\omega) \in \mathrm{H}(\mathrm{div}, D)$ for almost every $\omega \in \Omega$. We restrict the uncertain initial condition to an uncertain start location and maintain the shape of the function over all realizations. We refer to section 4.2 for the particular configuration and discretization.

3.3.4 The Acoustic Wave Equation under Uncertainty

Lastly, we recall the model problem 1.3 of an acoustic wave in first-order linear hyperbolic form in heterogeneous (unknown, and thereby randomly modeled) media. This model is of particular interest in seismic imaging and geophysics. Once again, we employ log-normal fields to model the material structure of the ground, i.e., we consider that the material density ρ in model (1.3) is randomly distributed and all other input data is deterministically given, particularly $\kappa(\mathbf{x}) \equiv 1$. Similar to the elliptic case, we define

$$\rho_{\min}(\omega) \coloneqq \min_{\mathbf{x} \in \overline{D}} \rho(\omega, \mathbf{x}) \quad \text{and} \quad \rho_{\max}(\omega) \coloneqq \max_{\mathbf{x} \in \overline{D}} \rho(\omega, \mathbf{x}).$$

By lemma 3.10, we know that a sample for a fixed $\omega \in \Omega$ satisfies $\rho(\omega, \cdot) \in C^t(\overline{D}) \subset \mathrm{L}^\infty(D)$. Hence, the mass operator $M(\omega)$ and the flux operator $A(\omega)$ as constructed in (2.18) satisfy the assumption 3.30 where the randomness in $\underline{B}_d(\omega)$ for $d = 1, \dots, \mathbf{d}$ is inherited from material. Furthermore, we can conclude that the material satisfies $c(\omega, \cdot) = \sqrt{\kappa_K / \rho_K} < \infty$ cell-wise which is important for the usage of the discrete model 2.36. By using the circulant embedding method to generate realizations of $\rho(\omega, \cdot)$ and thus of $c(\omega, \cdot)$ we can bound the error made by this method again by lemma 3.28. We refer to [185, section 2.3] for further analysis on the hyperbolicity of the problem. Once again, we omit the particular description of discretization and refer to section 4.3 where this is done.

3.4 Non-intrusive Estimators

Having outlined the properties of the uncertainty of the model problems in section 3.3 and the discretization techniques for the deterministic PDEs in chapter 2, we are now able to approximate the quantities of interest (QoI) with *non-intrusive* estimators. Non-intrusive estimators are very popular for UQ as they provide a natural separation of the spatio-temporal discretization and the stochastic simulation, i.e., we can employ statistical estimators on a set of discretized solutions computed on samples of the input data. We recall that we denote the approximate solution by u_ℓ and the corresponding QoI by Q_ℓ. The samples are indexed with m, i.e., $u_\ell(\omega^{(m)})$ is the discretized solution computed with the input data corresponding to $\omega^{(m)}$. However, by the FDNA 3.15 we can express samples of the input data by the vector $\mathbf{y}^{(m)} \in \mathbb{R}^K$, such that we occasionally write $u_\ell(\mathbf{y}^{(m)})$ and $\mathrm{Q}_\ell(\mathbf{y}^{(m)})$ or simply $u_\ell^{(m)}$ and $\mathrm{Q}_\ell^{(m)}$.

As mentioned in the introduction (chapter 1.3), the UQ problems considered here boil down to evaluating statistical moment integrals. In most interesting applications, it is impossible to evaluate these integrals and in practice they are approximated by a quadrature rule instead, i.e., we use the word quadrature rule and estimator for the same thing. Within this section, we introduce the three most popular non-intrusive estimators, the Monte Carlo method (MC), the quasi-Monte Carlo method (QMC) and the stochastic collocation (SC) method. Furthermore, we explain at the end of the section how these methods can be employed on a parallel computer and with multiple, subsequent rounds of estimation.

We interpret these methods in the following as quadrature rules for the expected value to underline again the non-intrusive nature of these methods as function evaluations for each quadrature point are computed independently.

$$\mathbb{E}[\mathrm{Q}] = \int_\Omega \mathrm{Q}(\omega)\mathrm{d}\mathbb{P} \approx \begin{cases} \widehat{\mathrm{Q}}^{\mathrm{MC}}_{\ell,M} & := M^{-1}\sum_{m=1}^{M} \mathrm{Q}_\ell(\mathbf{y}^{(m)}), \quad \mathbf{y}^{(m)} \in \Xi \subset \mathbb{R}^K \text{ (pseudo) randomly drawn} \\ \widehat{\mathrm{Q}}^{\mathrm{QMC}}_{\ell,M} & := M^{-1}\sum_{m=1}^{M} \mathrm{Q}_\ell(\mathbf{y}^{(m)}), \quad \mathbf{y}^{(m)} \in \Xi \subset \mathbb{R}^K \text{ deterministically selected} \\ \widehat{\mathrm{Q}}^{\mathrm{SC}}_{\ell,M} & := \sum_{m=1}^{M} w_m \mathrm{Q}_\ell(\mathbf{y}^{(m)}), \quad \mathbf{y}^{(m)} \in \Xi \subset \mathbb{R}^K \text{ taken from sparse grids} \end{cases} \tag{3.9}$$

Questions arising with the usage of these formulas are:

1. How does the dimension K of the integration domain influence the behavior of the method?
2. What is the class of integrands for which the method delivers approximations of high quality?
3. How many quadrature points does the method need to achieve an error tolerance of ϵ?
4. How expensive is the evaluation of a single sample $\mathrm{Q}_\ell(\mathbf{y}^{(m)})$ and what is the error?
5. How are the quadrature weights and points determined?

To tackle the last question first, MC methods draw their samples (pseudo) randomly (confer 3.2.1), QMC methods select the points deterministically e.g. by lattice rules, and SC methods use sparse or tensor grids (confer 3.2.4) with appropriate weights as input. By the FDNA 3.15, the points in the quadrature rules belong to a finite-dimensional space. However, they differ in the way they are constructed and how the function evaluations are weighted. The upcoming paragraphs will outline their difference.

For the fourth question, we assume throughout the section that we can bound the cost and the error of evaluating $\mathrm{Q}_\ell(\mathbf{y}^{(m)})$ with respect to ℓ. This evaluation requires solving a PDE with some approximation scheme like FEM and computing a QoI. The assumption acts as an interface to the previous chapter about FEM 2. Formally it is expressed as follows:

Assumption 3.35. For any realization of the input data $\mathbf{y}^{(m)}$ (any quadrature point), we assume that the discretization error $\mathrm{err}^{(m)}_{\mathrm{disc}}$ and the cost of the approximation scheme $\mathrm{C}^{(m)}_\ell$ are bounded w.r.t. the discretization parameter h_ℓ, i.e., that for some $\alpha > 0$ and $\gamma > 0$ it holds

$$\left|\mathrm{Q}(\mathbf{y}^{(m)}) - \mathrm{Q}_\ell(\mathbf{y}^{(m)})\right| \lesssim h_\ell^\alpha \qquad \text{and} \qquad \mathrm{C}^{(m)}_\ell := \mathrm{C}_\ell\left(\mathrm{Q}_\ell(\mathbf{y}^{(m)})\right) \lesssim h_\ell^{-\gamma}. \tag{3.10}$$

Definition 3.36. We denote by $\mathrm{C}_\epsilon(\widehat{\mathrm{Q}}_\ell)$ the *ϵ-cost* of an estimator, which is the computational cost to achieve a root mean squared error (RMSE) of ϵ, or a mean squared error (MSE) of ϵ^2, respectively (confer definition A.12). Measuring this cost can for example be done by:

1. Measuring the amount of floating point operations needed to achieve an error tolerance of ϵ.
2. Measuring the computing time to achieve an error tolerance of ϵ.

3.4.1 Monte Carlo Method

Since the Monte Carlo (MC) method has received enough introduction within this thesis already, we use this section to work out a few details of the method. The MC method is nothing but a sample mean, i.e., the weights in (3.9) are simply determined by $w_m = M^{-1}$ for $m = 1, \dots, M$. Assuming the QoI Q is approximated on some level ℓ, the mean squared error (MSE) (confer definition A.12) of this approach can be decomposed as in the following lemma.

Lemma 3.37. *The mean squared error (MSE) of the FEM-MC is given by*

$$\mathrm{err}_{\mathrm{MSE}}\left(\widehat{\mathrm{Q}}^{\mathrm{MC}}_{\ell,M}\right) = \mathbb{E}\left[\left(\widehat{\mathrm{Q}}^{\mathrm{MC}}_{\ell,M} - \mathbb{E}[\mathrm{Q}]\right)^2\right] = \underbrace{M^{-1}\mathbb{V}[\mathrm{Q}_\ell]}_{\textit{estimator error}} + \underbrace{\mathbb{E}[\mathrm{Q}_\ell - \mathrm{Q}]^2}_{\textit{FEM error / estimator bias}} .$$

Proof. A similar result will be shown in lemma 3.46. □

Theorem 3.38 (Bounded ϵ-cost of the MC method)**.** *Suppose assumption 3.35 is fulfilled with some positive rates $\alpha, \gamma > 0$ and some hidden constants independent of h_ℓ. Then for any $\epsilon > 0$, there exists a level $\ell \in \mathbb{N}$ and a number of samples M, such that*

$$\mathrm{err}_{\mathrm{RMSE}}\left(\widehat{\mathrm{Q}}^{\mathrm{MC}}_{\ell,M}\right) < \epsilon \qquad \textit{with} \qquad \mathrm{C}_\epsilon(\widehat{\mathrm{Q}}^{\mathrm{MC}}_{\ell,M}) \lesssim \epsilon^{-2-\frac{\gamma}{\alpha}} \tag{3.11}$$

Proof. A sufficient condition to achieve a RMSE of accuracy $\epsilon > 0$ is that the estimator variance and the estimator bias both are less or equal than $\epsilon^2/2$. By choosing $M = \mathcal{O}(\epsilon^{-2})$ and ℓ such that $|\mathbb{E}[\mathrm{Q}_\ell - \mathrm{Q}]| \lesssim h_\ell^\alpha = \mathcal{O}(\epsilon)$ by (3.10), this results with the cost estimate $\mathrm{C} = \mathcal{O}(M \cdot h_\ell^{-\gamma})$ in

$$\mathrm{C}_\epsilon(\widehat{\mathrm{Q}}^{\mathrm{MC}}_{\ell,M}) \lesssim \epsilon^{-2-\frac{\gamma}{\alpha}}.$$

□

3.4.2 Quasi-Monte Carlo Method

The formula for quasi-Monte Carlo quadrature equals the one of Monte Carlo, but the quadrature points are chosen deterministically rather than pseudo randomly. For the following explanations we restrict ourselves to the K-dimensional unit hypercube, i.e., to $\Omega = [0, 1]^K$ in (3.9). We follow the survey [114] and the book [10, section 3] but keep the explanation lean as QMC is not the main method of interest in this thesis. We are interested in the error

$$\epsilon = \left| \int_{[0,1]^K} \mathrm{Q}(\omega) \mathrm{d}\mathbb{P} - \widehat{\mathrm{Q}}^{\mathrm{QMC}}_{\ell,M} \right| \quad \text{with } \widehat{\mathrm{Q}}^{\mathrm{QMC}}_{\ell,M} \text{ defined in (3.9)}, \tag{3.12}$$

where the quadrature points are generated with *lattice rules* (a somewhat regular point distribution in Ω) or *nets* (e.g. Sobol points [228]). To derive an estimate for the error we will neglect the discretization with respect to ℓ for now. To evaluate how well a point set fills the hypercube Ω, the following two definitions are useful.

Definition 3.39. For $0 \leq a_k < b_k \leq 1$ with $k = 1, \dots, K$ and a set of points $\{\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(M)}\}$, we call

$$D_M^{(*)} = \sup_{A \in \mathcal{A}} \left| \frac{\left|\{\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(M)}\} \cap A\right|}{M} - |A| \right|, \quad |A| \hat{=} \text{Lebesgue measure of } A$$

ordinary discrepancy D_M, if $\mathcal{A}$ is the collection of all rectangles in $[0,1)^K$ of the form $\prod_{k=1}^K [a_k, b_k)$ and *star discrepancy* D_M^*, if $\mathcal{A}$ the collection of $\prod_{k=1}^K [0, b_k)$.

By this definition, the discrepancy of $\{\mathbf{y}^{(m)}\}_{m=1}^M$ is low if the proportion of points in an arbitrary subset A is close the proportional Lebesgue measure of A. It is well known that D_M^* grows for any sequence at least with $(\log M)^K/M$. Sequences with this growth behavior are often called *low-discrepancy* sequence.

Definition 3.40. Following the definition in [51], if Q is a sufficiently smooth function on $[0,1]^K$, we define the *Hardy-Krause variation* recursively by

$$V^{\text{HF}}(\text{Q}) := \int_{[0,1]^K} \left| \frac{\partial^k \text{Q}}{\partial y_1 \dots \partial y_K} \right| \mathrm{d}y_1 \dots \mathrm{d}y_K + \sum_{k=1}^K V^{\text{HF}}(\text{Q}_{y_k=1})$$

where $Q_{y_k=1}$ is the restriction of the function Q to the boundary $y_k = 1$.

Theorem 3.41 (Koksma-Hlawka). *For any sequence $\{\mathbf{y}^{(m)}\}_{m=1}^M$ and any function Q with a bounded Hardy-Krause variation, the integration error ϵ from 3.12 is bounded as*

$$\epsilon \leq V^{\text{HF}}(\text{Q}) D_M^*.$$

Proof. Given in [51, section 5.4]. □

Remark 3.42. We follow the remarks of [51, 10]

a) The computational domain Ω can be easily changed by a suitable transformation $\Omega \to [0,1]^K$.

b) The Koksma-Hlawka bound is a worst-case bound. In practice, the convergence is much faster than theoretically predicted.

c) Using theorem 3.41 and assuming that V^{HF} is finite, it is clear that the effort to achieve an error of ϵ is $\mathcal{O}((\log M)^K/M)$. This also justifies that QMC is considered to converge with $\mathcal{O}(M^{-1})$, simply if $M \gg K$.

d) Another aspect of this theorem is that the right-hand side is hard to compute. Finding good approximations for the star discrepancy is possible with some computational effort. However, determining the Hardy-Krause variation is very difficult and in the case of Gaussian variables it is even unbounded. Randomized QMC can for example be employed to recover a finite variation. We refer to [118, 10] for details.

Lattice Rules and Nets. Lastly, we present in figure 3.4 an example of 512 pseudo randomly generated samples on the left, 512 samples generated with the low-discrepancy Halton sequence in the middle and 512 samples generated with the low-discrepancy Sobol sequence on the right on a two-dimensional domain

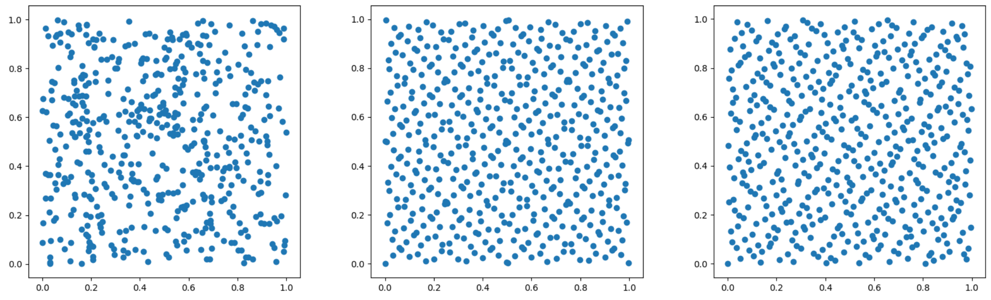


Figure 3.4: Two-dimensional pseudo-random number generation (left), Halton sequence (middle) and Sobol sequence (right). Each plot shows 512 data-points generated with `scipy.stats.qmc` [223]

(we refer to [199] for further details and omit a detailed description on the generation). We note that the low-discrepancy sequences cover the domain more evenly than the pseudo-random sequence.

3.4.3 Stochastic Collocation Method

Stochastic collocation (SC) is in some sense the most classical quadrature rule presented in this section. Even though it is used in (3.9) as a sampling based method, it is rather a high-dimensional cubature rule. To present an introduction on SC methods, we follow [265, 200, 12] and describe the method as an interpolation but remark that we have only used and implemented it as a quadrature rule for the expected value integral. For the sake of a compact representation, we restrict ourselves to tensor grids as introduced in 3.2.4 and refer to [200, 201] for an explanation using sparse grids.

We seek a numerical approximation to a solution of some PDE with uncertain input data in a finite-dimensional space $V_{\ell,\mathbf{p}}$ based on the tensor product $V_{\ell,\mathbf{p}} = \mathbb{Q}_{\mathbf{p}}(\mathbf{\Gamma}) \otimes V_\ell$. The space V_ℓ is some finite element space as described in section 2.2 and $\mathbb{Q}_{\mathbf{p}}(\mathbf{\Gamma})$ is the span of tensor product polynomials with degree at most $\mathbf{p} = (p_1, \dots, p_K)$, i.e., the space $\mathbb{Q}_{\mathbf{p}}(\mathbf{\Gamma}) = \bigotimes_{k=1}^K \mathbb{P}_{p_k}(\Gamma_k)$ with

$$\mathbb{P}_{p_k}(\Gamma_k) = \operatorname{span}\{\psi_{m_k},\, m_k = 0, \dots, p_k\} \quad \text{for} \quad k = 1, \dots, K.$$

Therefore, the dimension of $\mathbb{Q}_{\mathbf{p}}(\mathbf{\Gamma})$ is $M = \prod_{k=1}^K (p_k + 1)$. We construct the discrete solution $u_{\ell,\mathbf{p}} \in \mathbb{Q}_{\mathbf{p}}(\mathbf{\Gamma}) \otimes V_\ell$ by interpolating in the collocation points $\{\mathbf{y}^{(m)}\}_{m=1}^M$. To this end, we introduce two more things. First, we consider an auxiliary probability density function $\hat{f}_{\mathbf{Y}} \colon \Gamma \to \mathbb{R}^+$ to the true probability density function $f_{\mathbf{Y}} \colon \Gamma \to \mathbb{R}^+$, where

$$\hat{f}_{\mathbf{Y}}(\mathbf{y}) = \prod_{k=1}^K \hat{f}_{Y_k}(y_k) \quad \text{such that} \quad \left\| f_{\mathbf{Y}} / \hat{f}_{\mathbf{Y}} \right\|_{\mathrm{L}^\infty(\mathbf{\Gamma})} < \infty.$$

Second, we introduce for each dimension $k = 1, \dots, K$ a one-dimensional Lagrange polynomial basis $\{\psi_{m_k}\}_{m_k=0}^{p_k}$ on $\mathbb{P}_{p_k}(\Gamma_k)$ (confer example 2.4). With that we can define the basis $\{\boldsymbol{\psi}_m^{\mathbf{p}}(\mathbf{y})\}_{m=1}^M$ by a tensor product and write

$$u_{\ell,\mathbf{p}}(\mathbf{y}) = \sum_{m=1}^{M} u_\ell(\mathbf{y}^{(m)}) \psi_m^{\mathbf{p}}(\mathbf{y}).$$

Finally, for any continuous function $\mathrm{Q}\colon \boldsymbol{\Gamma} \to \mathbb{R}$, we can approximate the expected value integral with $\mathbb{E}[\mathrm{Q}] \approx \int_{\boldsymbol{\Gamma}} \mathrm{Q}(\mathbf{y}) \hat{f}_{\mathbf{Y}}(\mathbf{y}) \mathrm{d}\mathbf{y}$ by

$$\mathbb{E}[\mathrm{Q}] \approx \widehat{\mathrm{Q}}_{\ell,M}^{\mathrm{SC}} := \sum_{m=1}^{M} w_m \mathrm{Q}_\ell(\mathbf{y}^{(m)}) \quad \text{with} \quad w_m = \prod_{k=1}^{K} w_{m_k} \quad \text{and} \quad w_{m_k} = \int_{\Gamma_k} \psi_{m_k}^2(y) \hat{f}_{Y_k}(y) \mathrm{d}y.$$

Remark 3.43. a) Stochastic collocation originates from deterministic numerical methods for ODEs where the governing equation is interpolated in collocation points.

b) As in standard interpolation, stochastic collocation is more robust, if the collocation points are clustered at the boundary of the domain. Otherwise, this can lead to the well-known Runge phenomena [131]. A good set of collocation points can be achieved for example by using Chebyshev nodes.

c) The error analysis for the SC method is highly problem dependent. We therefore only restate shortly the estimates presented in [92]

$$\left|u - u_M^{\mathrm{tensor}}\right| \lesssim V^{\mathrm{tensor}}(u) M^{-\alpha/K}, \qquad \left|u - u_M^{\mathrm{sparse}}\right| \lesssim V^{\mathrm{sparse}}(u) (\log M)^K M^{-\beta},$$

where α and β depend on the differentiability of u with respect to $\mathbf{y}$. Hence, u has to be in a certain Sobolev space with mixed higher order derivatives to derive the above error estimate for a particular problem. The above result also makes clear that the usage of sparse grids over tensorized grids mitigates the curse of dimensionality.

d) The SC method usually requires for low-dimensional problems significantly less function evaluations than the MC or even the QMC method if the solution has enough regularity w.r.t. $\mathbf{y}$. Unfortunately, as seen in the previous remark, it is also very prone to high-dimensional problems since the amount of grid points grows rapidly with K. Hence, the SC method is applicable in practice if the KL expansion converges fast enough.

3.4.4 Parallel Non-intrusive Estimators

Last but not least, we introduce formulas which can handle the incremental and the parallel accumulation of higher-order central moments (confer definition A.11). The four key features these formulas have to provide are numerical stability, computability by a single pass through the data, parallelizability, and weighted updatability.

Standard approaches to compute higher-order central moments (including the variance, the skewness and the kurtosis) require passing through the data twice: first to compute the mean and then to compute the central moment. To make the matter worse, these computations can also result in numerical instabilities.

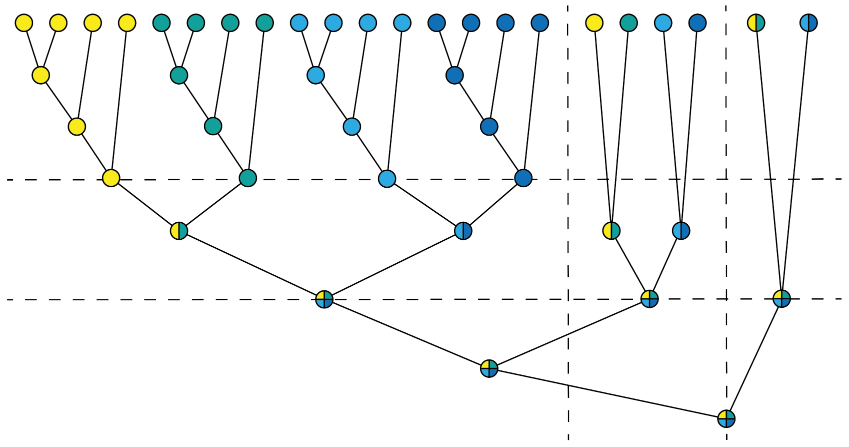


Figure 3.5: Illustration of the *triple-layered Welford update* on four parallel processes (each color represents one process) with three estimator rounds as binary-tree: Starting from the top of the illustration, the first layer until the dashed line is an incremental update by adding single samples to the accumulated statistics on each process. The second layer is the parallel update across all processes by recursively joining the sample statistics of the first layer. The last layer is the accumulation of the data over the estimator rounds. We remark that each round can be parallelized differently depending on the ration of the demanded samples in the round $M_{\mathfrak{i}}$ and the total amount of available processes $|\mathcal{P}|$ by (2.4).

Consider the task of estimating the variance of a random variable Y by observing finitely many samples M

$$\widehat{Y} = M^{-1} \sum_{m=1}^{M} y^{(m)} \mathrel{\widehat{=}} \text{sample mean}, \quad s_Y^2 = (M-1)^{-1} \underbrace{\sum_{m=1}^{M} (y^{(m)} - \widehat{Y})^2}_{=:S_{Y,2}} \mathrel{\widehat{=}} \text{sample variance}.$$

Here, the computation of $y^{(m)} - \widehat{Y}$ can get unstable if $y^{(m)} \approx \widehat{Y}$ for many samples, due to the phenomena of cancellation of floating point numbers, e.g. described in [137]. As we use the sample mean in the sample variance, we use the usual Bessel's correction in order to get an unbiased estimator for the variance [86, section 9.2]. With a slight correction proposed in [52] for $S_{Y,2}$ the stability can be improved by

$$S_{Y,2} = \sum_{m=1}^{M} (y^{(m)} - \widehat{Y})^2 - \underbrace{M^{-1} \left(\sum_{m=1}^{M} (y^{(m)} - \widehat{Y}) \right)^2}_{=0, \text{ if exact arithmetic is used}}.$$

However, this method is still based on a two-pass algorithm and does not provide incremental or parallel updates. In [255] by Welford and later in [52] by Chan et al. methods are proposed in order to compute the variance incrementally or in parallel. Furthermore, the generalization to an arbitrary central moment with weighted updates is described in [206], which serves as the main reference for this paragraph. To formalize this goal, we search for a way to estimate the q-th central moment (confer A.11 f)) with sample statistics $s_Y^q \approx \mathbb{M}_Y^q \coloneqq \mathbb{E}[(Y - \mu_Y)^q]$ incrementally and in parallel.

Triple-layered Welford Update. Welford's online algorithm 3.44, or rather its generalization, inspects each sample (or collection of samples) once and accumulates the estimates as the simulation runs. Thus, the method presented below provides all four required features and was taken form [206].

Method 3.44. Given two data sets $(W_i, \widehat{Y}_i, S_{i,2}, S_{i,3}, S_{i,4})_{i\in A,B}$, the accumulation of the sample statistics $\widehat{Y}, s_Y^2, s_Y^3, s_Y^4$ is done with

$$\begin{aligned}
W_{AB} &\leftarrow W_B + W_A \\
\delta_{AB} &\leftarrow \widehat{Y}_{W_B} - \widehat{Y}_{W_A} \\
\widehat{Y}_{W_{AB}} &\leftarrow \widehat{Y}_{W_A} + \tfrac{W_B}{W_{AB}}\delta_{AB} \\
S_{Y,2} &\leftarrow S_{A,2} + S_{B,2} + \tfrac{W_A W_B}{W_{AB}}\delta_{AB}^2 \\
S_{Y,3} &\leftarrow S_{A,3} + S_{B,3} + W_A\left(\tfrac{-W_B}{W_{AB}}\delta_{AB}\right)^3 + W_B\left(\tfrac{W_A}{W_{AB}}\delta_{AB}\right)^3 \\
&\quad + 3\delta_{AB}\left(S_{A,2}\left(\tfrac{-W_B}{W_{AB}}\right) + S_{B,2}\left(\tfrac{W_A}{W_{AB}}\right)\right) \\
S_{Y,4} &\leftarrow S_{A,4} + S_{B,4} + W_A\left(\tfrac{-W_B}{W_{AB}}\delta_{AB}\right)^4 + W_B\left(\tfrac{W_A}{W_{AB}}\delta_{AB}\right)^4 \\
&\quad + 4\delta_{AB}\left(S_{A,3}\left(\tfrac{-W_B}{W_{AB}}\right) + S_{B,3}\left(\tfrac{W_A}{W_{AB}}\right)\right) + 6\delta_{AB}^2\left(S_{A,2}\left(\tfrac{-W_B}{W_{AB}}\right)^2 + S_{B,2}\left(\tfrac{W_A}{W_{AB}}\right)^2\right) \\
s_Y^2 &\leftarrow (W_{AB}-1)^{-1} S_{Y,2} \\
s_Y^3 &\leftarrow \frac{\sqrt{W_{AB}}S_{Y,3}}{S_{Y,2}^{3/2}} \\
s_Y^4 &\leftarrow \frac{W_{AB}S_{Y,4}}{S_{Y,2}^2}
\end{aligned}$$

This method is used in three different ways. First, it is used to perform an incremental update of the statistical quantities on each process. In this case, the input to the method simplifies the weights to $W_A = W_{AB} - 1$, $W_B = 1$, $\widehat{Y}_A = \widehat{Y}_{\text{old}}$, $\widehat{Y}_B = y_{\text{new}}$. Second, the method is used to perform a parallel update across processes. The method simplifies in this case with $W_A = W_B$. And last, the same method is used to update the statistical quantities over several estimation rounds `i`. The implementation of all that is given in {37}. We refer to the upcoming section 3.5.3 where the need of this will become more clear. To give a visual intuition to the different update procedures we refer to figure 3.5.

3.5 Multi-level Estimators

We will now describe the main methods of this thesis – multi-level estimators. The section starts by recalling the MLMC method as Giles described it in [109] and [110] in 3.5.1. Considering a sequence of decreasing mesh widths $h_\ell = h_0 2^{-\ell}$ for $\ell = 0, \dots, L$, it is well known that the application of this algorithm is beneficial, if the following three assumptions are satisfied.

Assumption 3.45. a) The approximation scheme which computes u_ℓ and Q_ℓ is convergent with the rate $\alpha > 0$, i.e.,

$$|\mathbb{E}[\mathrm{Q}_\ell - \mathrm{Q}]| \lesssim h_\ell^\alpha \,. \tag{3.13}$$

b) Secondly, for some rate $\beta > 0$ the variance of $\mathrm{Y}_\ell \coloneqq \mathrm{Q}_\ell - \mathrm{Q}_{\ell-1}$ for $\ell \geq 1$ decays with

$$\mathbb{V}[\mathrm{Q}_\ell - \mathrm{Q}_{\ell-1}] \lesssim h_\ell^\beta \,. \tag{3.14}$$

c) Lastly, the cost to compute one evaluation of $\mathrm{Y}_\ell(\mathbf{y}^{(m)})$ is bounded for all realizations of the input data $\mathbf{y}^{(m)}$ by

$$\mathrm{C}\left(\mathrm{Y}_\ell(\mathbf{y}^{(m)})\right) \lesssim h_\ell^{-\gamma} \,, \tag{3.15}$$

with some $\gamma > 0$. In the case of optimal solvers, we have $\gamma \approx \mathbf{d}$ in the time independent and $\gamma \approx \mathbf{d} + 1$ in the time dependent case under consideration of the CFL condition [141].

For example for the elliptic problems in [58, 25, 245, 53, 55], it was proven that with Lagrange finite elements the first two assumptions are satisfied and thereby, that the MLMC method is applicable with a solver satisfying (3.15). However, as the imposed problems get more challenging, proving these assumptions a priori becomes harder, too. Thus, we want to continue this section in 3.5.2 by picking up on the ideas summarized in [108], where with an improved implementation of the method, these assumptions can be verified numerically on-the-fly as the simulation runs. Thereby, this approach is an adaptive algorithm finding its way to invest computational resources, such that a given RMSE error tolerance of $\epsilon > 0$ is reached.

A further practical improvement was proposed in [61] by the continuation MLMC (CMLMC) method, where the authors use a decreasing sequence of tolerances $\{\epsilon_{\mathtt{i}}\}_{\mathtt{i}\in\mathbb{N}_0}$ with $\epsilon_{\mathtt{i}} \to \epsilon_{\min} > 0$ in order to create a problem hierarchy. The problem hierarchy can then be exploited to get better on-the-fly estimations.

However, as we will describe in further detail, presetting an RMSE error tolerance $\epsilon > 0$ or a sequence of tolerances $\{\epsilon_{\mathtt{i}}\}_{\mathtt{i}\in\mathbb{N}_0}$ with $\epsilon_{\mathtt{i}} \to \epsilon_{\min} > 0$ and letting the algorithm run until the tolerance is undercut, has its disadvantages in the application due to the lack of sharp a priori knowledge about the total computational cost. Therefore, we propose a budgeted MLMC (BMLMC) method where we impose a computational budget as a constraint, resulting in a *knapsack* problem. We will combine ideas from the CMLMC method with techniques from dynamic programming (DP) to solve the knapsack problem. The key idea is to establish a subsequent problem structure and to find the overall solution after multiple estimation rounds reusing the results from previous iterations. We want to close section 3.5.3 by giving an outlook on other multilevel estimators like MLSC, MLQMC and MIMC and if they can be realized with a cost budget as well. Having a budgeted version of all these algorithms gives an empirically rigorous way to study their computational efficiency.

The proposed method is highly adaptive and thereby requires a dynamic load distribution. Hence, we will discuss the application and the properties of the parallelization approach proposed in chapter 2 on FEM in combination with the BMLMC method at the end of this section in 3.5.4.

3.5.1 Introduction to the Multi-level Monte Carlo Method

The underlying idea of the MLMC method is to construct a model hierarchy for the imposed problem. This is done by a sequence of nested meshes $\{\mathcal{M}_\ell\}_{\ell=0}^L$ with decreasing mesh widths $h_\ell = h_0 2^{-\ell}$ for $\ell = 0, \dots, L$. The goal is to avoid evaluating the model on the finest level as much as possible and to minimize the overall estimator variance. For a fixed finest level L, the expected value of Q_L can be written as a telescoping sum over the levels

$$\mathbb{E}[\mathrm{Q}_L] = \mathbb{E}[\mathrm{Q}_0] + \sum_{\ell=1}^{L} \mathbb{E}[\mathrm{Q}_\ell - \mathrm{Q}_{\ell-1}] = \sum_{\ell=0}^{L} \mathbb{E}[\mathrm{Y}_\ell], \qquad \mathrm{Y}_0 \coloneqq \mathrm{Q}_0, \qquad \mathrm{Y}_\ell \coloneqq \mathrm{Q}_\ell - \mathrm{Q}_{\ell-1}. \tag{3.16}$$

Each expected value of Y_ℓ in the telescoping sum is now estimated individually with a MC method (or another non-intrusive estimator, see section 3.4), resulting in the MLMC estimator

$$\widehat{\mathrm{Q}}^{\mathrm{MLMC}}_{\{M_\ell\}_{\ell=0}^L} = \sum_{\ell=0}^{L} \widehat{\mathrm{Y}}^{\mathrm{MC}}_{\ell, M_\ell} = \sum_{\ell=0}^{L} M_\ell^{-1} \sum_{m=1}^{M_\ell} \mathrm{Y}_\ell(\mathbf{y}^{(m)}) \,, \tag{3.17}$$

where $\{M_\ell\}_{\ell=0}^L$ denotes a sequence for the number of samples on each level. It is important that every $\mathrm{Y}_\ell(\mathbf{y}^{(m)}) = \mathrm{Q}_\ell(\mathbf{y}^{(m)}) - \mathrm{Q}_{\ell-1}(\mathbf{y}^{(m)})$ uses the same sample $\mathbf{y}^{(m)} \in \Omega$ for two different meshes. Since all the expected values $\mathbb{E}[\mathrm{Y}_\ell]$ are estimated independently, the variance of the MLMC method can be quantified on each level individually and with this, we obtain the following result for the mean squared error (MSE).

Lemma 3.46. *The mean squared error (MSE) of the FEM-MLMC is given by*

$$\mathrm{err}_{\mathrm{MSE}}\left(\widehat{\mathrm{Q}}^{\mathrm{MLMC}}_{\{M_\ell\}_{\ell=0}^L}\right) = \underbrace{\sum_{\ell=0}^{L} M_\ell^{-1} \mathbb{V}[\mathrm{Y}_\ell]}_{\textit{estimator variance}} + \underbrace{(\mathbb{E}[\mathrm{Q}_L - \mathrm{Q}])^2}_{\textit{FEM error / estimator bias}} . \tag{3.18}$$

Proof. Follows directly with $\mathbb{V}[\widehat{\mathrm{Q}}^{\mathrm{MLMC}}_{\{M_\ell\}_{\ell=0}^L}] = \sum_{\ell=0}^{L} M_\ell^{-1} \mathbb{V}[\mathrm{Y}_\ell]$ and $\mathbb{E}[\widehat{\mathrm{Q}}^{\mathrm{MLMC}}_{\{M_\ell\}_{\ell=0}^L}] = \mathbb{E}[\mathrm{Q}_L]$

$$\begin{aligned}
\mathrm{err}_{\mathrm{MSE}}\left(\widehat{\mathrm{Q}}^{\mathrm{MLMC}}_{\{M_\ell\}_{\ell=0}^L}\right) &= \mathbb{E}\left[\left(\widehat{\mathrm{Q}}^{\mathrm{MLMC}}_{\{M_\ell\}_{\ell=0}^L} - \mathbb{E}[\mathrm{Q}]\right)^2\right] \\
&= \mathbb{E}\left[\left(\widehat{\mathrm{Q}}^{\mathrm{MLMC}}_{\{M_\ell\}_{\ell=0}^L} - \mathbb{E}[\widehat{\mathrm{Q}}^{\mathrm{MLMC}}_{\{M_\ell\}_{\ell=0}^L}] + \mathbb{E}[\widehat{\mathrm{Q}}^{\mathrm{MLMC}}_{\{M_\ell\}_{\ell=0}^L}] - \mathbb{E}[\mathrm{Q}]\right)^2\right] \\
&= \mathbb{E}\left[\left(\widehat{\mathrm{Q}}^{\mathrm{MLMC}}_{\{M_\ell\}_{\ell=0}^L} - \mathbb{E}[\widehat{\mathrm{Q}}^{\mathrm{MLMC}}_{\{M_\ell\}_{\ell=0}^L}]\right)^2\right] + \left(\mathbb{E}[\widehat{\mathrm{Q}}^{\mathrm{MLMC}}_{\{M_\ell\}_{\ell=0}^L}] - \mathbb{E}[\mathrm{Q}]\right)^2 \\
&= \underbrace{\sum_{\ell=0}^{L} M_\ell^{-1} \mathbb{V}[Y_\ell]}_{\text{estimator variance}} + \underbrace{(\mathbb{E}[\mathrm{Q}_L - \mathrm{Q}])^2}_{\text{FEM error / estimator bias}} .
\end{aligned}$$

□

Assuming we want to achieve a MSE of ϵ^2, i.e., an RMSE tolerance of ϵ, we can reach this accuracy for

$\theta \in (0,1)$, if

$$(\mathbb{E}[\mathrm{Q}_L - \mathrm{Q}])^2 < (1-\theta)\epsilon^2 \quad \text{and} \quad \mathbb{V}[\widehat{\mathrm{Q}}^{\mathrm{MLMC}}_{\{M_\ell\}_{\ell=0}^L}] < \theta\epsilon^2.$$

The parameter θ can thereby be used to tune this variance bias trade-off (confer definition A.12) in order to favor the minimization of one term over the other. We will investigate this parameter θ in further detail in the numerical experiments in chapter 4.

Remark 3.47. Comparing this result with the MSE in lemma 3.37, we can qualitatively argue for the cost reduction because of the following two reasons (confer [58]):

a) Assumption (3.14) and the MSE in (3.18) imply that fewer samples have to be drawn the finer the level ℓ to achieve a MSE tolerance of ϵ^2.

b) No matter which MSE tolerance ϵ^2 is targeted, the cost to compute a single sample on the coarsest level does not grow with $\epsilon \to 0$. However, the coarsest useful mesh width h_0 depends on the application. In particular, the coarsest mesh has to be able to resolve features of the solution and the input data sufficiently enough in order to contribute to the multi-level simulation in a computationally beneficial way. We will observe this phenomena later in chapter 4.

Quantitatively, the computational cost of the method is given by

$$\mathrm{C}(\widehat{\mathrm{Q}}^{\mathrm{MLMC}}_{\{M_\ell\}_{\ell=0}^L}) = \sum_{\ell=0}^{L}\sum_{m=1}^{M_\ell} \mathrm{C}_\ell(\mathbf{y}^{(m)}) = \sum_{\ell=0}^{L} M_\ell M_\ell^{-1} \sum_{m=1}^{M_\ell} \mathrm{C}_\ell(\mathbf{y}^{(m)}) = \sum_{\ell=0}^{L} M_\ell \widehat{\mathrm{C}}_\ell, \tag{3.19}$$

where $\widehat{\mathrm{C}}_\ell$ is the sample mean of the cost and $\mathrm{C}_\ell(\mathbf{y}^{(m)})$ is the cost of an individual sample on level ℓ. Following [108], we now want to find the optimal sequence of samples $\{M_\ell\}_{\ell=0}^L$, such that estimator cost is minimized while achieving an MSE tolerance of ϵ^2. By presetting the MSE tolerance, we can also deduce from (3.13) the highest level L. Thereby we choose L such that the estimator bias is smaller than $(1-\theta)\epsilon^2$. Hence, we know ϵ and L and can deduce the optimal choice of samples as shown in the following lemma.

Lemma 3.48. *For a given ϵ and a given L, the optimal number of samples for the MLMC estimator on each level is given by*

$$M_\ell^{\mathrm{opt}} = \left\lceil \left(\sqrt{\theta}\epsilon\right)^{-2} \sqrt{\frac{\mathbb{V}[\mathrm{Y}_\ell]}{\widehat{\mathrm{C}}_\ell}} \left(\sum_{\ell'=0}^{L} \sqrt{\mathbb{V}[\mathrm{Y}_{\ell'}]\widehat{\mathrm{C}}_{\ell'}}\right)\right\rceil \quad \textit{for} \quad \ell = 0,\dots,L \tag{3.20}$$

with the ceiling function $\lceil\cdot\rceil : \mathbb{R} \to \mathbb{N}$.

Proof. We minimize the cost and fix the estimator variance to $\theta\epsilon^2$, i.e., we search for a solution to

$$\begin{aligned} \min_{\{M_\ell\}_{\ell=0}^L} \quad & \mathrm{C}(\widehat{\mathrm{Q}}^{\mathrm{MLMC}}_{\{M_\ell\}_{\ell=0}^L}) = \sum_{\ell=0}^{L} M_\ell \widehat{\mathrm{C}}_\ell \\ \text{s.t.} \quad & \sum_{\ell=0}^{L} M_\ell^{-1}\mathbb{V}[\mathrm{Y}_\ell] = \theta\epsilon^2. \end{aligned}$$

Regarding M_ℓ for $\ell = 0, \dots, L$ as continuous variables, we have the Lagrange function

$$\mathcal{L}(M_0, \dots, M_L, \nu) = \sum_{\ell=0}^{L} M_\ell \widehat{\mathrm{C}}_\ell + \nu^2 \left(\sum_{\ell=0}^{L} M_\ell^{-1} \mathbb{V}[\mathrm{Y}_\ell] - \theta \epsilon^2 \right)$$

giving the conditions

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial M_\ell} &= \widehat{\mathrm{C}}_\ell - \nu^2 \frac{\mathbb{V}[\mathrm{Y}_\ell]}{M_\ell^2} \overset{!}{=} 0 \quad &\iff \quad & M_\ell = \nu \sqrt{\frac{\mathbb{V}[\mathrm{Y}_\ell]}{\widehat{\mathrm{C}}_\ell}} \\ \frac{\partial \mathcal{L}}{\partial \nu} &= 2\nu \left(\sum_{\ell=0}^{L} M_\ell^{-1} \mathbb{V}[\mathrm{Y}_\ell] - \theta \epsilon^2 \right) \overset{!}{=} 0 \quad &\iff \quad & \sum_{\ell=0}^{L} M_\ell^{-1} \mathbb{V}[\mathrm{Y}_\ell] = \theta \epsilon^2 . \end{aligned}$$

Hence, by inserting the first into the second condition, we get

$$\nu = \left(\sqrt{\theta} \epsilon \right)^{-2} \sum_{\ell=0}^{L} \sqrt{\mathbb{V}[\mathrm{Y}_\ell] \widehat{\mathrm{C}}_\ell},$$

which gives (3.20) by accounting for the fact that M_ℓ^{opt} must be an integer.

□

Remark 3.49. Instead of looking at the RMSE as tolerance, the authors of [61, 208] proposed to look for a small *estimator tolerance* (confer definition A.12) with high probability. Hence, with some small probability $\delta \in (0,1)$, we look for

$$\mathbb{P}\left(\left| \widehat{\mathrm{Q}}^{\mathrm{MLMC}}_{\{M_\ell\}_{\ell=0}^{L}} - \mathbb{E}[\mathrm{Q}] \right| > \epsilon \right) \le \delta \quad \iff \quad \mathbb{P}\left(\left| \widehat{\mathrm{Q}}^{\mathrm{MLMC}}_{\{M_\ell\}_{\ell=0}^{L}} - \mathbb{E}[\mathrm{Q}] \right| \le \epsilon \right) > 1 - \delta$$

with the confidence $1-\delta$. The optimal number of samples (3.20) gets an additional factor determined by δ. We refer to [208] for further details and stick to the notation where ϵ is the target RMSE tolerance.

With adapted notation, the initial theorem of [110] is given in [58] and states the following.

Theorem 3.50 (Bounded ϵ-cost of the MLMC method)**.** *Suppose assumption 3.45 is fulfilled with some positive rates $\alpha, \beta, \gamma > 0$ with $\alpha \ge \frac{1}{2} \min\{\beta, \gamma\}$ and some hidden constants independent of h_ℓ. Then for any $0 < \epsilon < \mathrm{e}^{-1}$, there exists a maximum level $L \in \mathbb{N}$ and a sequence of samples $\{M_\ell\}_{\ell=0}^{L}$, such that*

$$\mathrm{err}_{\mathrm{RMSE}} \left(\widehat{\mathrm{Q}}^{\mathrm{MLMC}}_{\{M_\ell\}_{\ell=0}^{L}} \right) < \epsilon \qquad \textit{with} \qquad \mathrm{C}_\epsilon(\widehat{\mathrm{Q}}^{\mathrm{MLMC}}_{\{M_\ell\}_{\ell=0}^{L}}) \lesssim \begin{cases} \epsilon^{-2} & \beta > \gamma \\ \epsilon^{-2} \log(\epsilon)^2 & \beta = \gamma \\ \epsilon^{-2-(\gamma-\beta)/\alpha} & \beta < \gamma \end{cases} . \tag{3.21}$$

Proof. A complete proof can be found in [58] in a notation similar to ours. □

Remark 3.51. We present the short discussion of the theorem following [108]. The theorem states that we have a theoretical bound for the computational cost determined by the target RMSE tolerance ϵ. By combining the assumptions 3.45 and the optimal choice of $\{M_\ell\}_{\ell=0}^L$ in (3.20), we get

$$M_\ell^{\text{opt}} \sim \sqrt{\frac{\mathbb{V}[\mathrm{Y}_\ell]}{\widehat{\mathrm{C}}_\ell}} \sim \sqrt{\frac{h_\ell^\beta}{h_\ell^{-\gamma}}} \quad \Rightarrow \quad M_\ell^{\text{opt}} \sim h_\ell^{(\beta+\gamma)/2} \tag{3.22}$$

with an appropriate choice for the constant and the highest level L, such that the estimator bias is smaller than $(1-\theta)\epsilon^2$. To get an estimate on the computational cost the proof of the theorem investigates the following: Does the cost grow faster than the variance decays with an increasing level or vise versa? To answer this question it is useful to look at the product $\mathbb{V}[\mathrm{Y}_\ell]\widehat{\mathrm{C}}_\ell$, since by (3.19) and (3.20), we know that $\mathrm{C}(\widehat{\mathrm{Q}}^{\text{MLMC}}_{\{M_\ell\}_{\ell=0}^L}) \lesssim \sum_{\ell=0}^L \sqrt{\mathbb{V}[\mathrm{Y}_\ell]\widehat{\mathrm{C}}_\ell}$.

$\beta > \gamma$: In this case the cost is dominated by $\sqrt{\mathbb{V}[\mathrm{Y}_0]\widehat{\mathrm{C}}_0}$ and thus largest on the lowest level, where $M_0 \sim \epsilon^{-2}$ samples are needed to achieve the MSE tolerance of ϵ.

$\beta = \gamma$: In this case the cost is spread almost evenly across all levels. We omit the details and refer to [58] since this case is irrelevant in practice.

$\beta < \gamma$: The dominant cost is on the highest level, (3.13) implies that $h_L^\alpha = \mathcal{O}(\epsilon)$ and with the best case of $\beta = 2\alpha$ (since $\mathbb{V}[\mathrm{Y}_\ell]$ is similar in magnitude to $\mathbb{E}[\mathrm{Y}_\ell^2] \geq (\mathbb{E}[\mathrm{Y}_\ell])^2$), it follows for the cost that $\mathrm{C}_L = \mathcal{O}(\epsilon^{-\gamma/\alpha})$. Thereby, the cost of the method is dictated by the computational complexity of solving one deterministic version of the problem assuming sufficiently smooth noise.

Lastly, we remark that in any case the computational savings compared to simple Monte Carlo $\mathrm{C}_\epsilon(\widehat{\mathrm{Q}}^{\text{MC}}_{\ell,M}) \lesssim \epsilon^{-2-\gamma/\alpha}$ (confer equation (3.11)) are significant.

3.5.2 Implementation of the Multi-level Monte Carlo Method

In the previous paragraph, we have explored the variance reduction and by that, the computational benefits of the MLMC method. However, the challenge in using the MLMC method is to show that the assumptions of theorem 3.50 are actually fulfilled. In [108], techniques for the implementation of the algorithm are summarized. This includes a verification of the assumptions as well as a way to find the optimal sequence $\{M_\ell\}_{\ell=0}^L$ and the highest level L on-the-fly. As a start, we recognize that equation (3.20) contains the variance $\mathbb{V}[\mathrm{Y}_\ell]$ and the sample mean of the cost $\widehat{\mathrm{C}}_\ell$, which are unknown a priori. The implementation idea is to run the MLMC method with an initial number of levels and samples $\{M_\ell^{\text{init}}\}_{\ell=0}^{L_{\text{init}}}$ to get first guesses for the sample mean of the cost $\widehat{\mathrm{C}}_\ell$ and the variance $\mathbb{V}[\mathrm{Y}_\ell]$ by the sample variance $s^2_{\mathrm{Y}_\ell}$. With the initial estimates on $\widehat{\mathrm{C}}_\ell$ and $s^2_{\mathrm{Y}_\ell}$, the MLMC method is executed until the target RMSE ϵ is reached. Since the samples from the initial round do not have to be computed again, the required sample amount ΔM_ℓ is introduced by

$$\Delta M_\ell := \max\left\{\widehat{M}_\ell^{\text{opt}} - M_\ell, 0\right\} \quad \text{with} \quad \widehat{M}_\ell^{\text{opt}} = \left\lceil \left(\sqrt{\theta}\epsilon\right)^{-2} \sqrt{\frac{s^2_{\mathrm{Y}_\ell}}{\widehat{\mathrm{C}}_\ell}} \left(\sum_{\ell'=0}^L \sqrt{s^2_{\mathrm{Y}_{\ell'}}\widehat{\mathrm{C}}_{\ell'}}\right)\right\rceil, \tag{3.23}$$

where $\widehat{M}_\ell^{\text{opt}}$ is the optimal sample amount based on the existing data and M_ℓ is the amount of previously computed samples.

Furthermore, Richardson extrapolation can be used to actually find the highest level L to meet the required RMSE tolerance of ϵ (confer [108]). To do so, we see by $h_\ell = h_0 2^{-\ell}$, the simplification of assumption 3.13 as $|\mathbb{E}[\mathrm{Q} - \mathrm{Q}_\ell]| = ch_\ell^\alpha$ and $\mathbb{E}[\mathrm{Q} - \mathrm{Q}_L] = \mathbb{E}[\mathrm{Q} - \mathrm{Q}_{L-1}] - \mathbb{E}[\mathrm{Q}_L - \mathrm{Q}_{L-1}]$ that

$$ch_L^\alpha = ch_{L-1}^\alpha - \mathbb{E}[\mathrm{Q}_L - \mathrm{Q}_{L-1}] \qquad \Longleftrightarrow \qquad ch_L^\alpha = \frac{\mathbb{E}[\mathrm{Q}_L - \mathrm{Q}_{L-1}]}{(2^\alpha - 1)}.$$

From that, we can deduce an estimate of the bias by

$$\widehat{\mathrm{err}}_{\mathrm{disc}} = \max \left\{ \frac{\widehat{\mathrm{Y}}_\ell}{2^{\widehat{\alpha}} - 1} 2^{-\widehat{\alpha}(L-\ell)} : \ell = 1, \dots, L \right\}, \tag{3.24}$$

which also incorporates lower levels for robustness of the estimate [108] and uses an estimate $\widehat{\alpha}$ for α by fitting the data $\{\widehat{\mathrm{Y}}_\ell\}_{\ell=0}^L$ to assumption 3.45 with

$$\min_{(\widehat{\alpha},\, c)} \sum_{\ell=1}^{L} (\log_2 \widehat{\mathrm{Y}}_\ell + \widehat{\alpha}\ell - c)^2 \tag{3.25}$$

giving an the estimate $\widehat{\alpha}$ for the rate. This can be done in a similar fashion for γ and β.

Lastly, the estimator variance is approximated with the sample variance of Y_ℓ on each level

$$\widehat{\mathrm{err}}_{\mathrm{input}} = \sum_{\ell=0}^{L} M_\ell^{-1} s_{\mathrm{Y}_\ell}^2. \tag{3.26}$$

Hence, the MSE can be estimated with $\widehat{\mathrm{err}}_{\mathrm{MSE}} = \widehat{\mathrm{err}}_{\mathrm{input}} + \widehat{\mathrm{err}}_{\mathrm{disc}}^2$. Overall, this results in the following implementation of the method.

Method 3.52. Choose the initial estimation round $\{M_\ell^{\mathrm{init}}\}_{\ell=0}^L$, choose a target RMSE tolerance of $\epsilon > 0$, a splitting parameter $\theta \in (0, 1)$ and a set of processing units $\mathcal{P}$. The implementation of the MLMC method with on-the-fly estimation is then described by (the implementation in C++ is given in {38}):

```
function MLMC($\epsilon, \{M_\ell^{\mathrm{init}}\}_{\ell=0}^L$):
    $\{\Delta M_\ell\}_{\ell=0}^L \leftarrow \{M_\ell^{\mathrm{init}}\}_{\ell=0}^L$
    while $\widehat{\mathrm{err}}_{\mathrm{MSE}} \geq \epsilon^2$:
      for $\ell = L, \dots, 0$:
        if $\Delta M_\ell \geq 0$:                $\mathrm{C}_\ell, \mathrm{Q}_\ell, \mathrm{Y}_\ell \leftarrow$ MC($\Delta M_\ell$) $\leftarrow$ MSS($\Delta M_\ell, \mathcal{P}$) with (2.24)
      if $\widehat{\mathrm{err}}_{\mathrm{disc}}$(data) $\geq \sqrt{1-\theta}\epsilon$:   $L \leftarrow L + 1$ with (3.24)
      if $\widehat{\mathrm{err}}_{\mathrm{input}}$(data) $\geq \theta\epsilon^2$:      $\{\Delta M_\ell\}_{\ell=0}^L$ with (3.26) and (3.23)
      update $\widehat{\alpha}, \widehat{\beta}, \widehat{\gamma}$ with (3.25)
    return $\widehat{\mathrm{Q}}$ with (3.17)
```

Remark 3.53. We remark the inverted level loop which has benefits for the load distribution [16]. This topic will be explained in further detail at the end of the section. Furthermore, we note that the Monte Carlo `MC` method on level ℓ is just a wrapper for solving a multi-sample system `MSS` which we explained in section 2.4.3. This makes it very easy to replace the MC method by QMC or SC without writing too much new code and without rethinking the parallelization.

The above method is further improved by the continuation MLMC, as proposed in [61]. It was also applied in [208] for computational fluid dynamics (CFD) and in [167] for electromagnetic field scattering. As mentioned in the introduction of this section, a central idea is to create a sequence $\{\epsilon_i\}_{i\in\mathbb{N}_0}$ with $\epsilon_{\mathtt{i}} \to \epsilon_{\min}$. By doing so, a problem hierarchy is established, where each problem is tempered by its predecessor problems, resulting in robust on-the-fly estimations. We will incorporate this idea in our final method proposal - the budgeted MLMC method.

3.5.3 A Budgeted Multi-level Monte Carlo Method

We want to propose a new variant of the MLMC method, namely, the budgeted MLMC (BMLMC) method which we consider as a key contribution in this thesis. The goal is to replace the imposed target RMSE tolerance ϵ by a cost budget $\mathrm{B} > 0$ measured in $[\mathrm{B}] = \#\mathrm{CPU} \cdot \mathrm{seconds}$. This is motivated by two practical reasons.

- Firstly, it is common in HPC applications that a cost budget has to be reserved in order to queue a job on the HPC cluster. In our notation this is depicted by

$$\mathrm{B} = |\mathcal{P}| \cdot \mathrm{T}, \quad |\mathcal{P}| \mathrel{\widehat{=}} \text{total amount of processing units}, \quad \mathrm{T} \mathrel{\widehat{=}} \text{wall-clock computing time}.$$

  It requires a priori knowledge about the problem, the convergence rates, the parallelization and the computer to actually be able to tell which tolerance ϵ can be achieved with which budget B. If the computation exceeds the budget, the cluster will kill the job with unfinished work and potentially unusable intermediate results. If the method undershoots the budget, it was probably queued longer than it has to. Thus, replacing the preassigned tolerance ϵ by a preassigned budget B and letting the method find the smallest ϵ, while exhausting the budget, is the more natural requirement in HPC applications.

- Secondly, having a budgeted algorithm lets us answer the question raised in the introduction, namely, which algorithm stack should be used, given a certain problem? The answer to this question is: the one which gives the smallest error, given that all algorithm combinations used the same computational resources B. Thereby, we can study this question now empirically, simply by trying out different algorithms with an equal budget B and comparing the achieved error estimates. This idea perfectly fits to the continuous delivery (CD) workflow we explored in section 2.5.2. In particular, the benchmarking stage finds the best algorithm combination in a fully automated manner under cost restriction.

If the final method is capable of exhausting the computational budget without overshoot, we call a particular run *feasible*. Furthermore, to develop the final method, we need to change the point of view on theorem 3.50.

Theorem 3.54 (Convergence of the BMLMC method)**.** *For a feasible run of the BMLMC method, we have an error estimate w.r.t. the computational budget*

$$\epsilon \lesssim \begin{cases} \mathrm{B}^{-1/2} & \beta > \gamma \\ \mathrm{B}^{-\alpha/(2\alpha+(\gamma-\beta))} & \beta < \gamma \end{cases}. \tag{3.27}$$

Proof. If the budget is exhausted, i.e., in the case of a feasible run, the final ϵ-cost equals the budget. Thereby, the error estimate simply follows by inverting (3.21). Similar results can be found in [185] and [156] where they used the word *work* instead of budget. □

This result does not tell us how to utilize the budget. Thereby, the new challenge for the algorithm is to find the best way to invest B, such that we effectively minimize the error. Formally, this is expressed by a *knapsack* problem:

Knapsack Problem for MLMC. Find L and $\{M_\ell\}_{\ell=0}^L$, such that the MSE is minimized while staying within the cost budget B, i.e.,

$$\min_{(L,\{M_\ell\}_{\ell=0}^L)} \quad \mathrm{err}_{\mathrm{MSE}} = \sum_{\ell=0}^{L} M_\ell^{-1} \mathbb{V}[\mathrm{Y}_\ell] + (\mathbb{E}[\mathrm{Q}_L - \mathrm{Q}])^2 \tag{3.28}$$

$$\text{s.t.} \quad \sum_{\ell=0}^{L} \sum_{m=1}^{M_\ell} \mathrm{C}_\ell(\mathbf{y}^{(m)}) \leq \mathrm{B}. \tag{3.29}$$

We remark the change of perspective compared to lemma 3.48. The objective now is to minimize the complete MSE including the bias while the constraint is given by the computational budget.

The typical toy problem of this class reads as follows: Find a collection of items, such that it does not exceed a weight limit and that it maximizes the value carried by a knapsack. Knapsack problems, in general, are combinatorial optimization problems and often arise while searching for the optimal allocation of resources. They have been studied extensively in the last decades as the integer optimization problem is NP-hard[3] requiring efficient algorithms [50], and they appear in a broad range of applications in manufacturing, computer networks and financial modeling [45].

Clearly in our context, the items are the samples and the cost of each sample represents the weight. The objective function is the error minimization which is a nonlinear and convex function with respect to B by (3.27). Hence, we are in the case described in [45, section 4]. A popular way to approximate solutions to knapsack problems are algorithms designed with the dynamic programming technique [45, 50].

[3]A decision problem, i.e., a problem that has a binary solution, is of the class of *NP* problems, if the solution can be verified, but not necessarily found, in polynomial time. NP stands for *nondeterministic polynomial-time*.

A problem is of the class NP-*hard*, if it is at least as hard as the hardest problems in NP, i.e., every problem in NP can be reduced in polynomial time to the NP-hard problem.

Dynamic Programming. Dynamic programming (DP) is a design technique for algorithms to solve optimization problems. The key idea is to split up the initial problem into subproblems and solve them recursively, while *memoizing* and reusing preexisting results. It was developed in the 1940s and 1950s by Richard Bellman, who coined the term with the explanation that the word *dynamic* is associated with the incremental treatment of the subproblems and *programming* is another word for optimal planning (we refer to his autobiography [78], but also to his non-mathematical interpretation of dynamic programming in [32]). Ever since, the field has gone through much development and finds incredibly many algorithmic applications like finding the shortest path in route planning, finding the optimal designs in engineering or assembly-line scheduling in manufacturing [34]. We refer to [63, section 15.3] which is a well established introduction to this algorithm design pattern. Here, the intent is to shortly summarize enough material to design the final method with dynamic programming techniques. Generally speaking, dynamic programming algorithms satisfy the following:

- Algorithms designed with dynamic programming split up the initially imposed problem in *overlapping and acyclic subproblems*. This can be done for example by expressing the objective value V in a recursive function of *value-returns* or *pay-offs*.
- Dynamic programming is based on *memoization*, i.e., the reutilization of preexisting computations. The idea is to create a data structure with which multiple computations of the same result are avoided and the solutions to subsequent problems are stored. Thereby, dynamic programming is often characterized as *careful brute-force*.
- Based on some *optimal policy* considering the *state* s represented by the data structure, a *guess* of, or a *decision* for, the optimal next step is executed subject to some constraint. We denote this with $a \in \mathcal{A}$ where $\mathcal{A}$ denotes the set of admissible decisions. Therefore, the best decision to make depends on previous solutions to the subproblems. This makes dynamic programming a multi-stage planning and a sequential decision-making process.
- The runtime of dynamic programming often follows (quasi)-polynomial time with respect the number of problems (quasi-polynomial, if the cost of the subproblems grows).

Remark 3.55. Reinforcement learning is closely related to dynamic programming, however, the terminology is slightly different. In particular, the program is called *agent* which collects data about its environment and chooses its *actions* getting some *reward*. We further remark that in a stochastic environment dynamic programming is a Markov decision process and refer to the well-cited survey [146] for an introduction to reinforcement learning.

Definition 3.56. a) A sequence of decisions satisfies the *optimality condition*, if based on any initial state s, the sequence follows an optimal policy with respect to each subsequent state. In the literature, this is also often called *principle of optimality* or *Bellman principle*. As a consequence, dynamic programming finds a solution to a problem by dividing it into subproblems, where each is solved optimally with respect to its given state.

b) A *Bellman equation* is the formalization of the *optimality condition* and expresses the expected optimal value V as a recursive function of the current reward depending on the state s and the chosen action $a \in \mathcal{A}$, and the optimal value of the subsequent state s' which is chosen by action a, i.e.,

$$\mathrm{V}(s) = \max_{a \in \mathcal{A}} \{\mathrm{R}(a,s) + \gamma \mathrm{V}(s')\} \quad \text{with} \quad \mathrm{R}(a,s) \mathrel{\hat{=}} \text{reward function},$$

where $\gamma \in (0,1]$ is the discount factor which will not be important for our investigations. Depending on the problem, the Bellman equation might look rather different. However, its defining feature is the expression of the objective value at a given state as a recursive function. Therefore, the Bellman equation is useful to evaluate the value of a state by considering the current and subsequent rewards, provided only optimal actions are taken.

We now turn to the final method development. As a start, we recognize once again that we do not know the exact quantities in (3.28) and (3.29), but rather we have to consider the approximations (3.26) and (3.24). The following optimization problem is still NP-hard but at least only contains computable quantities.

Approximated Knapsack Problem for MLMC: Find L and $\{M_\ell\}_{\ell=0}^L$, such that the estimated MSE is minimized, while staying within the cost budget B, i.e.,

$$\min_{(L,\{M_\ell\}_{\ell=0}^L)} \quad \widehat{\mathrm{err}}_{\mathrm{MSE}} = \sum_{\ell=0}^{L} M_\ell^{-1} s_{\mathrm{Y}_\ell}^2 + \left(\max \left\{ \frac{\widehat{\mathrm{Y}}_\ell}{2^{\widehat{\alpha}} - 1} 2^{-\widehat{\alpha}(L-\ell)} : \ell = 1, \dots, L \right\} \right)^2 \tag{3.30}$$

$$\text{s.t.} \quad \sum_{\ell=0}^{L} M_\ell \widehat{\mathrm{C}}_\ell \leq \mathrm{B} \tag{3.31}$$

Remark 3.57. By considering the above approximated knapsack problem we chose to *discretize first and optimize second*. The downside of this approach is that if $s_{\mathrm{Y}_\ell}^2$, $\widehat{\mathrm{C}}_\ell$, $\widehat{\mathrm{Y}}_\ell$ and $\widehat{\alpha}$ are inaccurate, the optimization delivers poor results, too. However, by using dynamic programming we will update these quantities continuously such that this disadvantage becomes less important.

The idea now is to solve the above problem by splitting it up into several estimation rounds with a decreasing sequence of tolerances $\{\epsilon_{\mathtt{i}}\}_{\mathtt{i} \in \mathbb{N}_0}$ as in the CMLMC method [61] to create subsequent optimization problems. To this end, we equip all quantities of section 3.5.2 with an index $\mathtt{i}$. In the following, we motivate the existence of a *pay-off/reward* function depending on the chosen action $\epsilon_{\mathtt{i}}$, and the current state of the simulation, i.e., the collected data up to $\mathtt{i} - 1$.

Suppose we are in estimation round $\mathtt{i}$ and $M_{0,\ell}^{\mathrm{init}}$ is given. For $\ell = 0, \dots, L_{\mathtt{i}}$, we separate

$$M_{\mathtt{i},\ell} = \underbrace{M_{\mathtt{i}-1,\ell}}_{\text{available data}} + \underbrace{\Delta M_{\mathtt{i},\ell}}_{\text{optimal choice}} \quad \text{with} \quad \Delta M_{0,\ell} := M_{0,\ell}^{\mathrm{init}} \quad \text{and} \quad M_{-1,\ell} := 0$$

such that $\Delta M_{\mathtt{i},\ell}$ follows (3.23), hence after the estimation round we have $M_{\mathtt{i},\ell} = \widehat{M}_{\mathtt{i},\ell}^{\mathrm{opt}}$. Thereby, we can express the amount of samples based on the currently available data and some *optimal policy*, i.e., $\Delta M_{\mathtt{i},\ell}$ is chosen such that the cost is minimized and a MSE tolerance of $\epsilon_{\mathtt{i}}^2$ is reached. With $\Delta M_{\mathtt{i},\ell}$ and method 3.44, we further separate

$$\widehat{\mathrm{C}}_{\mathtt{i},\ell} = \widehat{\mathrm{C}}_{\mathtt{i}-1,\ell} + \tfrac{\Delta M_{\mathtt{i},\ell}}{\widehat{M}^{\mathrm{opt}}_{\mathtt{i},\ell}}(\Delta\widehat{\mathrm{C}}_{\mathtt{i},\ell} - \widehat{\mathrm{C}}_{\mathtt{i}-1,\ell}) \quad \text{with} \quad \Delta\widehat{\mathrm{C}}_{\mathtt{i},\ell} \coloneqq \tfrac{1}{\Delta M_{\mathtt{i},\ell}} \textstyle\sum_{m=M_{\mathtt{i}-1,\ell}+1}^{\widehat{M}^{\mathrm{opt}}_{\mathtt{i},\ell}} \mathrm{C}_\ell^{(m)}$$

$$\widehat{\mathrm{Y}}_{\mathtt{i},\ell} = \widehat{\mathrm{Y}}_{\mathtt{i}-1,\ell} + \tfrac{\Delta M_{\mathtt{i},\ell}}{\widehat{M}^{\mathrm{opt}}_{\mathtt{i},\ell}}(\Delta\widehat{\mathrm{Y}}_{\mathtt{i},\ell} - \widehat{\mathrm{Y}}_{\mathtt{i}-1,\ell}) \quad \text{with} \quad \Delta\widehat{\mathrm{Y}}_{\mathtt{i},\ell} \coloneqq \tfrac{1}{\Delta M_{\mathtt{i},\ell}} \textstyle\sum_{m=M_{\mathtt{i}-1,\ell}+1}^{\widehat{M}^{\mathrm{opt}}_{\mathtt{i},\ell}} \mathrm{Y}_\ell^{(m)}$$

and likewise for

$$s^2_{\mathrm{Y}_{\mathtt{i},\ell}} = (\widehat{M}^{\mathrm{opt}}_{\mathtt{i},\ell} - 1)^{-1} \sum_{m=1}^{\widehat{M}^{\mathrm{opt}}_{\mathtt{i},\ell}} (\mathrm{Y}_\ell^{(m)} - \widehat{\mathrm{Y}}_{\mathtt{i},\ell})^2,$$

we separate with method 3.44

$$S_{\mathrm{Y}_{2,\mathtt{i},\ell}} = S_{\mathrm{Y}_{2,\mathtt{i}-1,\ell}} + \Delta S_{\mathrm{Y}_{2,\mathtt{i},\ell}} + \tfrac{M_{\mathtt{i}-1,\ell}\Delta M_{\mathtt{i},\ell}}{\widehat{M}^{\mathrm{opt}}_{\mathtt{i},\ell}}(\widehat{\mathrm{Y}}_{\mathtt{i}-1,\ell} - \Delta\widehat{\mathrm{Y}}_{\mathtt{i},\ell})^2,$$

where

$$\Delta S_{\mathrm{Y}_{2,\mathtt{i},\ell}} = \textstyle\sum_{m=M_{\mathtt{i}-1,\ell}+1}^{\widehat{M}^{\mathrm{opt}}_{\mathtt{i},\ell}} \left(\mathrm{Y}_\ell^{(m)} - \Delta\widehat{\mathrm{Y}}_{\mathtt{i},\ell}\right)^2.$$

As $\widehat{\alpha}_{\mathtt{i}}$ is computed with a fit to the available data (3.25), we can express (3.30) and (3.31) for a particular estimation round $\mathtt{i}$ as a nonlinear function of preexisting data (the *state*) and the *optimal policy* (3.23). This motivates a function for the *pay-off* purely determined by the state and the optimal policy. We denote this function by $\Delta\mathrm{err}_{\mathrm{MSE}}(\{\Delta M_{\mathtt{i},\ell}\}_{\ell=0}^{L}, \mathtt{data}[\mathtt{i}-1])$ which represents the error reduction in one estimation round, if $\{\Delta M_{\mathtt{i},\ell}\}_{\ell=0}^{L}$ samples are additionally computed. We further define $\mathrm{B}_{\mathtt{i}}$ as the left-over budget in round $\mathtt{i}$ and denote with $\mathrm{B}_0 \coloneqq \mathrm{B}$ the initially imposed budget. By (3.23) we see that the amount of samples is actually guided by $\epsilon_{\mathtt{i}}$. Thereby, with $\epsilon_{\mathtt{i}}$ as hyperparameter and with the cost prediction $\widehat{\mathrm{C}}_{\mathtt{i}} = \sum_{\ell=0}^{L} \Delta M_{\mathtt{i},\ell}\widehat{\mathrm{C}}_{\mathtt{i}-1,\ell}$, the Bellman equation for finding the solution to the approximated knapsack problem (3.30) and (3.31) can be expressed with

$$\mathrm{V}(\mathrm{B}_{\mathtt{i}}, \epsilon_{\mathtt{i}}) = \max_{\substack{\{\Delta M_{\mathtt{i},\ell}\}_{\ell=0}^{L_{\mathtt{i}}} \\ \text{s.t. } \widehat{\mathrm{C}}_{\mathtt{i}} < \mathrm{B}_{\mathtt{i}}}} \left\{ \Delta\mathrm{err}_{\mathrm{MSE}}\left(\{\Delta M_{\mathtt{i},\ell}\}_{\ell=0}^{L_{\mathtt{i}}}, \mathtt{data}[\mathtt{i}-1]\right) + \mathrm{V}\left(\mathrm{B}_{\mathtt{i}} - \sum_{\ell=0}^{L_{\mathtt{i}}} \mathrm{C}_{\mathtt{i},\ell},\, \eta \cdot \epsilon_{\mathtt{i}}\right)\right\}, \tag{3.32}$$

where the value of the initial round is given by

$$\mathrm{V}^{\mathrm{init}}(\mathrm{B}_0, \{M_\ell^{\mathrm{init}}\}_{\ell=0}^{L_0}) = \widehat{\mathrm{err}}_{\mathrm{MSE}}(\{M_\ell^{\mathrm{init}}\}_{\ell=0}^{L_0}) - \mathrm{V}\left(\mathrm{B}_0 - \sum_{\ell=0}^{L} \mathrm{C}_\ell,\, \eta \cdot \widehat{\mathrm{err}}_{\mathrm{MSE}}(\{M_\ell^{\mathrm{init}}\}_{\ell=0}^{L_0})\right)$$

and $\eta \in (0,1)$ is a chosen reduction factor determining how fast the sequence $\{\epsilon_{\mathtt{i}}\}_{\mathtt{i}\in\mathbb{N}_0}$ decays. So far, we have not discussed the minimization of the bias yet. If in (3.30) the bias becomes larger than $\sqrt{1-\theta}\epsilon_{\mathtt{i}}$ and if we have enough budget left, i.e., $\widehat{\mathrm{C}}_{\mathtt{i}} < \mathrm{B}_{\mathtt{i}}$, we draw additional samples on level $L+1$ and stop the optimization otherwise.

Concluded, function (3.32) is the expression of the subsequent minimization of the MSE under consideration of the cost budget. Collecting all of the above in a recursive algorithm we formulate the final BMLMC method.

Method 3.58. Choose the initial estimation round $\{M_\ell^{\text{init}}\}_{\ell=0}^{L_0}$, a cost budget $\mathrm{B} > 0$, a splitting parameter $\theta \in (0,1)$, a set of processing units $\mathcal{P}$ and the reduction factor $\eta \in (0,1)$. The implementation of the BMLMC is then described by:

```
data $= \Big\{\mathtt{i} \mapsto \Big\{\mathrm{err}_\mathtt{i}, \{M_{\mathtt{i},\ell}\}_{\ell=0}^{L_\mathtt{i}}, \{\widehat{\mathrm{C}}_{\mathtt{i},\ell}\}_{\ell=0}^{L_\mathtt{i}}, \{\widehat{\mathrm{Q}}_{\mathtt{i},\ell}\}_{\ell=0}^{L_\mathtt{i}}, \{\widehat{\mathrm{Y}}_{\mathtt{i},\ell}\}_{\ell=0}^{L_\mathtt{i}}, \dots\Big\}\Big\}$

function BMLMC($\mathrm{B}_0, \{M_{0,\ell}^{\text{init}}\}_{\ell=0}^{L_0}$):
    for $\ell = L_0, \dots, 0$:    $\mathrm{C}_\ell, \mathrm{Q}_\ell, \mathrm{Y}_\ell \leftarrow$ MC($M_{0,\ell}^{\text{init}}$) $\leftarrow$ MSS($M_{0,\ell}^{\text{init}}, \mathcal{P}$)
    Update data[0]    return BMLMC($\mathrm{B}_0 - \sum_{\ell=0}^{L_\mathtt{i}} \mathrm{C}_\ell, \eta \cdot \mathrm{err}_0$)

function BMLMC($\mathrm{B}_\mathtt{i}, \epsilon_\mathtt{i}$):
    if $\mathrm{B}_\mathtt{i} \approx 0$:    return $\mathrm{err}_{\mathtt{i}-1}$
    if $\widehat{\mathrm{err}}_{\mathtt{disc}}$(data) $\geq \sqrt{1-\theta}\epsilon_\mathtt{i}$:    $L_\mathtt{i} \leftarrow L_\mathtt{i} + 1$
    if $\widehat{\mathrm{err}}_{\mathtt{input}}$(data) $\geq \theta\epsilon_\mathtt{i}^2$:    $\widehat{M}_{\mathtt{i},\ell}^{\text{opt}} \leftarrow \left\lceil \left(\sqrt{\theta}\epsilon_\mathtt{i}\right)^{-2} \sqrt{\frac{s^2_{\mathrm{Y}_{\mathtt{i}-1,\ell}}}{\widehat{\mathrm{C}}_{\mathtt{i}-1,\ell}}} \left(\sum_{\ell'=0}^{L_\mathtt{i}} \sqrt{s^2_{\mathrm{Y}_{\mathtt{i}-1,\ell'}} \widehat{\mathrm{C}}_{\mathtt{i}-1,\ell'}}\right) \right\rceil$
    $\{\Delta M_{\mathtt{i},\ell}\}_{\ell=0}^{L_\mathtt{i}} \leftarrow \max\Big\{\widehat{M}_{\mathtt{i},\ell}^{\text{opt}} - M_{\mathtt{i}-1,\ell}, 0\Big\}$
    $\widehat{\mathrm{C}}_\mathtt{i} \leftarrow \sum_{\ell=0}^{L_\mathtt{i}} \Delta M_{\mathtt{i},\ell} \widehat{\mathrm{C}}_{\mathtt{i}-1,\ell}$
    if $\widehat{\mathrm{C}}_\mathtt{i} = 0$:    return BMLMC($\mathrm{B}_\mathtt{i}, \eta \cdot \epsilon_\mathtt{i}$)
    if $\widehat{\mathrm{C}}_\mathtt{i} > \mathrm{B}_\mathtt{i}$:    return BMLMC($\mathrm{B}_\mathtt{i}, 0.5 \cdot (\epsilon_\mathtt{i} + \epsilon_{\mathtt{i}-1})$)
    for $\ell = L_\mathtt{i}, \dots, 0$:    $\mathrm{C}_\ell, \mathrm{Q}_\ell, \mathrm{Y}_\ell \leftarrow$ MC($\Delta M_{\mathtt{i},\ell}$) $\leftarrow$ MSS($\Delta M_{\mathtt{i},\ell}, \mathcal{P}$)
    Update data[i]    return BMLMC($\mathrm{B}_\mathtt{i} - \sum_{\ell=0}^{L_\mathtt{i}} \mathrm{C}_\ell, \epsilon_\mathtt{i}$)
```

The implementation in C++ is given in {39}.

In order to better understand the mechanisms of this method, we refer to the discussion at the beginning of chapter 4 outlining the different paths the algorithm can take.

Remark 3.59. The actual implementation in C++ is not done with a recursive function but in an equivalent formulation with a while-loop. This is also often called a *bottom up* implementation which has the advantage over the recursive implementation (*top down*) to avoid an increased memory consumption on the stack. However, the recursive formulation is easier to derive mathematically.

Lemma 3.60. *The BMLMC method is MSE-consistent with respect to the computational budget* B.

Proof. First we realize that with an infinite budget (3.29) we have an unrestricted optimization problem. Then we can deduce by (3.13), (3.32) and (3.24) that we will be able to draw infinitely many levels, and therefore we can conclude for the MLMC estimator realized in a budgeted implementation that

$$\lim_{\mathrm{B}\to\infty} \mathbb{E}\left[\widehat{\mathrm{Q}}^{\mathrm{MLMC}}_{\{M_\ell\}_{\ell=0}^L}\right] = \lim_{\mathrm{B}\to\infty} \mathbb{E}\left[\sum_{\ell=0}^{L} M_{\mathtt{i},\ell}^{-1} \sum_{m=1}^{M_{\mathtt{i},\ell}} \mathrm{Y}_\ell(\mathbf{y}^{(m)})\right] = \lim_{\mathrm{B}\to\infty} \sum_{\ell=0}^{L} M_{\mathtt{i},\ell}^{-1} \sum_{m=1}^{M_{\mathtt{i},\ell}} \mu_{\mathrm{Y}_\ell}$$

$$= \lim_{\mathrm{B}\to\infty} \sum_{\ell=0}^{L} \mu_{\mathrm{Y}_\ell} = \lim_{\mathrm{B}\to\infty} \mu_{\mathrm{Q}_L} = \lim_{\mathrm{L}\to\infty} \mu_{\mathrm{Q}_L} = \mu_{\mathrm{Q}}.$$

□

Corollary 3.61. *For a fixed amount of processes $|\mathcal{P}|$, the theorem 3.54 and the lemma 3.60 also hold with respect to computational time* T.

Remark 3.62. We lastly discuss the application of this idea to multi-level stochastic collocation (MLSC) and multi-level quasi-Monte Carlo (MLQMC). As a start, we recognize that a key component of dynamic programming is the nested problem structure. With nested sparse grids and appropriate lattice rules for QMC this problem structure can be retained. However, this results in a less flexible choice of $\Delta M_{\mathtt{i},\ell}$ which has to be taken into account appropriately. We refer to the discussion in [231, section 3.5] or [245] on rounding strategies of sparse grids which is related to this problem. Furthermore, the on-the-fly estimation of all quantities and errors is not as straightforward as it is for simple Monte Carlo. Hence, randomized QMC and techniques like Romberg quadrature rules for nested sparse grids have to be employed to get suitable error estimators. As previously discussed in this thesis, realizing these methods in a budgeted framework would allow to truly find the best method combination for each problem.

3.5.4 Full Parallelization of the BMLMC Method

Here, the goal is to discuss the parallelization introduced in chapter 2 in combination with the BMLMC method. We present a broader discussion and classification on parallelization and follow the introduction of [126, 69, 37]. As a start, we have to ask ourselves why we should even parallelize. The simple reason is to speed up the computation, such that it is done in a reasonable amount of time, say "over lunch" or "over night" [126]. Often enough however, time is not the limiting factor but rather memory. If the problem is too big to fit on one machine, one has no other choice than to parallelize. The problems discussed in this thesis are huge and for the most interesting applications one computer is simply not big enough. Generally speaking, parallelization is a scalability problem such as they appear in manufacturing or traffic and often, such as in this thesis, parallelization is also a resource allocation problem. Many approaches can be taken to parallelize. Thereby, developing a parallelization strategy starts by identifying the type of parallelism which is employable.

- *Data Parallelism:* If multiple processes can work on different parts of the same data, we refer to this as single program multiple data (SPMD).
- *Functional Parallelism:* Big problems can sometimes be split into disparate subtasks working together by exchanging data and synchronizing. This is also often called multiple program multiple data (MPMD).

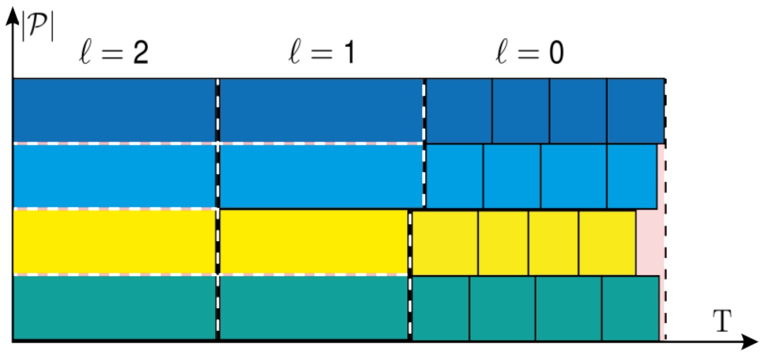


FIGURE 3.6: Load balancing in a single exemplary estimation round. Light red areas correspond to parallelization losses, other colors correspond to processes. The black dashed lines represent transitions between levels, the red dashed lines represent parallelization losses in the FEM system.

The parallelism we explained in section 2.1 is SPMD. This is the first distinguishing aspect of our approach to most other solutions which use MPMD by separating the UQ and the FEM side. Furthermore, we pick up the discussion in [79] and [16] to classify the parallelization particularly in the contex of multi-level estimators. For these methods there are three layers of possible parallel execution:

- *Solver parallelism:* The PDE solver is parallelized by domain decomposition. Even though this technique is well established in most advanced finite element tools, it has the significant drawback to require communication at the interfaces of the subdomains.
- *Sample parallelism:* The samples $\{\mathbf{y}^{(m)}\}_{m=1}^{M_\ell}$ on a fixed level ℓ are computed in parallel. This parallelism inherently provides independent computations except for some post-processing after the routine.
- *Level parallelism:* The estimators on the levels $\ell = 0, \dots, L$ are executed in parallel. This parallelism is also inherently independent but offers less opportunity to distribute the work than in the sample parallelism. The reason is that the sum over the levels usually contains way less terms than the sum over the samples.

To classify our implementation from section 2.1, we have a solver and sample parallelism and no level parallelism. The parallelism can be classified further:

- *Homogeneous parallelism:* The same number of processes are used for each sample, if they are computed on the same level.
- *Heterogeneous parallelism:* Samples on the same level can get assigned a different number of processes.

The proposed BMLMC method is homogeneous in each estimation round using the multi-sample system, but potentially heterogeneous over multiple estimation rounds (confer 3.5). This is due to the fact that the distribution is done *dynamically* for each new estimation. To classify the method as dynamic, we follow again [16, 79]:

Solver	Sample	Level	Homo.	Hetero.	Static	Dynamic	External	Integrated
MSS, [240, 238, 237, 16, 79]	MSS, [240, 238, 237, 16, 79]	[240, 238, 237, 16]	[240, 238, 79]	BMLMC, [237, 238, 79]	[237, 240, 238, 79]	BMLMC, [16, 79]	[237, 240, 238]	BMLMC, [16]

Table 3.1: Classification of parallelisms proposed in literature and in this thesis.

- *Static parallelism:* In a static scheduling of the computational work, the distribution is based on a priori assumptions about the cost on each level, e.g. this can be based on the size of the discretization scheme, but also be adapted to properties of the input data known before the sampling starts.
- *Dynamic parallelism:* In a dynamic scheduling the computational work is distributed on-the-fly and does not require a priori knowledge about the expected cost of the samples.

Another important aspect of the parallelism is where the synchronization of processes has to happen. Generally speaking, deciding for the synchronization points is a trade-off between potentially having many idling processes on the machine waiting for new tasks to be assigned to, and having a less optimal load distribution and oversampling because of poor statistical estimates on the cost and the variance. Lastly, we also propose the classification of the parallelism in external and integrated:

- *External parallelism:* We classify a parallelization as external, if it is based on an external software distributing multiple instances of PDE solvers on a computing cluster.
- *Integrated parallelism:* We classify an implementation as integrated, if the same software used to execute the PDE solver is used to distribute the samples, i.e., only one program instance is started working on distributed memory instead of multiple instances with different input data.

The last criteria is very technical and requires to look at the actual implementation of the method on the HPC system. However, the criteria can be seen as technical interpretation of SPMD and MPMD. In table 3.1, we have collected literature on different scheduling approaches and classified them according to the above criteria, where for the last one we have tried to consider the actual implementation, if publicly available.

Concluded, there are different techniques to distribute the computational load of multi-level estimators, each with its advantages but also with its challenges. The method we propose could be classified as a *dynamic and heterogeneous sample and solver parallelism in an integrated framework as SPMD.*

Figure 2.2 depicts the load distribution of an exemplary estimation round on three levels with $M_2 = 1, M_1 = 2$ and $M_0 = 16$. The sample on level two is processed on all four processes at first. Next, two samples on level one follow where each one is computed on two processes. Lastly sixteen samples on level zero are computed, where each processor handles four individual samples. The light red color correspond to parallelization losses due to either idling processes (red areas) or due to communication losses (dashed red lines, corresponding to communication across subdomains).

Remark 3.63. a) The best configuration for the parallelization parameters can be found yet again by looking at the achieved accuracy, given that each configuration has been using the same computational budget. We refer to chapter 4 on the numerical results for further details.

b) Within the scope of the above remark, the optimal amount of synchronization points and locations can be studied empirically. The BMLMC method does allow to tune the amount of synchronization points by adjusting η.

Performance Measurements

In order to evaluate the parallelization and quantify its scalability, we follow the books [126] and [69] and restate the following two performance measures.

Strong scaling: When we look for strong scaling in a parallel program, we try to reduce the overall runtime by adding further processing units. Thereby, when we try to measure the strong scaling of the parallelization, the problem size, e.g. the number of samples M in a Monte Carlo simulation, remains fixed, while the number of processing units is increased. Then, we look at the achieved speed up $\mathrm{T}_{\mathrm{min}}/\mathrm{T}_{\mathrm{parallel}}$, where $\mathrm{T}_{\mathrm{min}}$ is the computing time associated with the execution on $|\mathcal{P}_{\mathrm{min}}|$ processors and $\mathrm{T}_{\mathrm{parallel}}$ is the measured computing time achieved by employing $|\mathcal{P}| > |\mathcal{P}_{\mathrm{min}}|$ processing units. Often $|\mathcal{P}_{\mathrm{min}}| = 1$, however under certain circumstance, e.g. due to memory restrictions, multiple processors are needed to solve the problem at least on $|\mathcal{P}_{\mathrm{min}}|$ processors. The optimal speedup factor is then given by $|\mathcal{P}| \,/\, |\mathcal{P}_{\mathrm{min}}|$. For a standard Monte Carlo method this optimal speedup factor can almost be achieved. However, synchronization, communication overhead, unequal work loads and non-parallelizable parts of the algorithm reduce in almost every complex application the efficiency. *Amdahl's law* accounts for this by splitting the program in a sequential part of portion $(1-\lambda)$ and a parallel part of portion λ with $\lambda \in (0,1)$. By doing so, we can redefine the theoretical parallel execution time by

$$\mathrm{T}_{\mathrm{parallel}}^{\mathrm{Amdahl}} = (1-\lambda) + \frac{\lambda}{|\mathcal{P}|/|\mathcal{P}_{\mathrm{min}}|}, \qquad \lim_{|\mathcal{P}|\to\infty} \mathrm{T}_{\mathrm{parallel}}^{\mathrm{Amdahl}} = (1-\lambda). \tag{3.33}$$

Hence, adding infinitely many processing units does not infinitely speed up the code as the seriell parts of the program, like synchronization tasks of the algorithm or I/O operations on the hard-drive, simply do not scale. Concluded, for algorithms that scale strongly in parallel, we can answer the question: how much faster is the execution, if we use $|\mathcal{P}|$ processing units instead of $|\mathcal{P}_{\mathrm{min}}|$.

Weak scaling: In many practical applications, we look for programs that satisfy weak scaling since strong scaling, as for simple Monte Carlo, is unachievable. In this case the memory is the limiting resource, i.e., the problem size per computing unit has to remain the same, or at least stay below a hardware dependent maximum, while additional computing units are utilized to solve larger problems. A deterministic PDE solver distributing its domain on multiple cores is a good example for a weakly scaling parallelism. Strong scaling can not be expected here since using more cores just leads to a highly fragmented and distributed domain – in the extrem case, each process is assigned to one cell in the mesh. The communication overhead becomes dominant and program actually slows down. To utilize more processing units, the problem has to become bigger. Therefore, to measure weak scaling we try to keep the computational time fixed, i.e, $\mathrm{T}_{\mathrm{min}} \approx \mathrm{T}_{\mathrm{parallel}}$ while we increase the problem size. The increased problem size is then compensated with additional processes units.

We note at this point, that measuring weak scaling of a parallelized standard MLMC method is rather difficult. The reason is that keeping the computational time fixed while increasing the problem size alongside with the amount of processes most likely does not result in the optimal error reduction, i.e, weak scaling is measured, but we solve the wrong problem. For example, we could compute twice the amount of samples on each level, compensate the larger problem with a doubled amount of processing units and judge by the ration of the computational times $\mathrm{T}_{\mathrm{min}}/\mathrm{T}_{\mathrm{parallel}} \leq 1$ how well the program scales weakly. However, simply adding samples (or levels) to increase the problem size does not necessarily reduce the overall error. Hence, utilizing the BMLMC method is rather helpful to measure the weak scaling of the parallelization, i.e., we answer the question: how much is the error reduced, if we add further processing units? Since a smaller error directly corresponds to a larger problem, this is our way to measure the weak scaling of the parallelization. In particular, we consider theorem 3.54 in the case that $\beta > \gamma$ and conclude

$$2^e \overset{!}{=} \frac{\mathrm{err}_{2|\mathcal{P}|}}{\mathrm{err}_{|\mathcal{P}|}} \sim \frac{(2\,|\mathcal{P}| \cdot \mathrm{T})^{-1/2}}{(|\mathcal{P}| \cdot \mathrm{T})^{-1/2}} = 2^{-1/2}. \tag{3.34}$$

The closer the exponents, the better is the weak scaling of the parallelization.

3.6 The UQ-Extension to M++

This last section in this chapter is used to reflect on the developed UQ software and prepare the next chapter on the numerical results. We now have all mathematical tools at hand to actually be able to approximate solutions of PDEs with uncertain input data. However, we have not yet discussed the software tool which is capable of executing this task on a HPC cluster.

As the computer is developed, and as new mathematical ideas based upon the ability of the computer to carry out billions of simple instructions become widely known, we will see an elimination of the mathematical middleman through out many economic, engineering, and scientific domains. The mathematician will thus be a typical victim of automation and sophistication. - Richard E. Bellman, 1966 [32],

After over a decade of research on MLMC methods and numerous theoretical investigations, it is likely to expect that this method and its family will find many more applications, especially when it is utilized in a commercial and industrial context. Deriving sharp a priori convergence results for all applications which are about to come might be either impossible or an incredible effort for numerical analysts. Yet the method will show that it works in many practical applications as long as the assumptions 3.45 are fulfilled. Hence, it is fair to say that multi-level estimators are established tools for UQ and, to pick up on the above quote of Richard Bellman, we think it is time to "eliminate the mathematical middleman" by automating the investigations on these methods.

Many new challenges in UQ are already under investigation, for example stochastic optimal control problems [101, 100, 175, 124] or inverse UQ [139, 189, 136, 138, 74, 217, 73]. All of these new investigations will rely on highly efficient forward UQ in the application. Hence reliable, automated and integrated forward UQ will be a great help to focus on new theoretically interesting problems and make them useful in the application. Thereby, we propose an automated framework to conduct experiments and to falsify hypotheses to further establish the science of large-scale forward UQ with multi-level methods. The framework is capable to reproduce the theoretical achievements of many publications and to further extend the spectrum of possible investigations.

The remaining part of the section will describe the developed software on a high level and argue for some technical design choices we made, starting with the programming language and the parallelization technology. We follow the discussion of [69, section 14] and highlight our decisions.

Parallelization Technology. The technologies to distribute the computation can be divided in two classes. Shared memory technologies like OpenMP allow to execute parallel computations without worrying about the data access and exchange. This approach is highly performant yet has the disadvantage that, if the problem gets too big for one computer, it simply does not work anymore. On the other hand there are distributed memory technologies like OpenMPI. In this approach a distributed data structure has to be designed, however, the usage is not restricted to a single machine anymore and the code can be deployed on large scale computing clusters across several nodes. We chose to use OpenMPI since our problems are simply too big to fit on one machine. Furthermore, MPI provides the possibility to define subsets of processing units as communicators. We have made use of this feature extensively in order to realize the multi-sample systems described in section 2.4.3.

Programming Language. Generally speaking, every programming language has its advantages and its disadvantages. Hence, the programming language has to fit to the application of the software. Since we have a scientific HPC application, the choice to be made is between compiled languages like C or C++ offering highly performant and hardware near implementations, and interpreted languages like Python or Matlab which come with a large toolbox of numerical and statistical algorithms. The downside of developing C++ code with MPI is that a deep understanding of the parallel data structure, memory management and multicore processing has to be acquired to develop the system. In some sense, the extremely fast execution of the code comes with an increased development time. Before starting a HPC project, one should be aware of what is important, i.e., rapid code and model development or very fast code execution in production. Interpreted programming languages like Matlab or Python offer much shorter development times by utilizing a lot of preexisting computing routines and libraries and by providing the user leaner syntax. However, this comes with the cost of slower execution times due to layers of abstraction interpreted programming languages add, as well as less flexibility for special tasks like parallelization even though a lot of effort, see e.g. [247], is made to overcome this problem. Well established software libraries often provide both, fast execution on distributed memory via compiled languages in the backend plus lightweight wrapping for example written in Python to be accessible for a broader user base. We started our development with a mixed approach of Python, C++ and OpenMPI, however, it turned out to be poorly maintainable and extendable without native Python bindings to the C++ code. In the end, we settled on a pure C++/OpenMPI implementation which offers the highest performance, scalability and the cleanest extension to M++. We compensated the more approachable front end in Python by passing configuration files for the methods and problems to the Gitlab CI application programming interface (API) (confer section 2.5) and exploiting the semi-automated data-model cycle 2.5.

Hardware Architecture. Lastly, we have to discuss the hardware available on the HoReKa supercomputer and what was used by us. GPU clusters are gaining popularity since they utilize several thousands of low frequency cores processing large data arrays in parallel. Thereby, GPUs are popular in data-flow driven computations like machine learning. However, in computational sciences CPU-based architectures are still more common and established. We only used CPU-architectures but are planning to integrate the linear algebra library Ginkgo (confer example 2.45) in order to be able to deploy computations on GPUs as well. GPU based UQ seems very promising as the strengths of GPU hardware match the requirements of non-intrusive UQ algorithms.

Software Design of the Extension. Since we have already presented a detailed discussion on the FEM library in section 2.5, we now only present an overview on the modules for the UQ extension to M++.

`lib1_generators`: This part of the code corresponds to the theory of section 3.2, i.e., it uses the pseudo random number generator *SPRNG5* and the sparse grid generator *TASMANIAN* to provide realizations of random fields and stochastic processes generated by truncated KL expansions or the circulant embedding method.

`lib2_problems`: The second library builds upon the sample generators and defines the uncertain model problems. The corresponding theory is explained in section 3.3.

`lib3_pdesolvers`: The third layer in the code is basically a wrapper around M++, i.e., at this part of the software project the UQ and FEM side merge.

`lib4_estimators`: Last but not least, the estimator library utilizes all of the above. The theory behind this is explained in section 3.4 and 3.5.

Lastly, we close this software section by presenting a list of other UQ software tools, again without any claim that this is even remotely a complete list.

Example 3.64. a) *UQLab* is Matlab-based software framework for uncertainty quantification [174] used by over 4000 researchers and engineers and has been subject of many publications. We refer to the webseite for further details [252].

b) *ALSVID-UQ* [5] is a MLMC finite volume solver for hyperbolic conservation laws and was used in [237, 240, 238] and in one of our main sources [185].

c) In [17, 16] the software *FEMPAR* [91] was used. *FEMPAR* is a finite element software written in Fortran but has recently been extended with MLMC methods.

d) As previously mentioned, we used the sparse grid library *TASMANIAN* [234]. It provides a Matlab, Python and C++ interface and a well written user documentation [233].

e) *PyMC* [211] is a probabilistic programming library for Python that allows users to build Bayesian models with a simple Python API and fit them using Markov chain Monte Carlo (MCMC) methods.

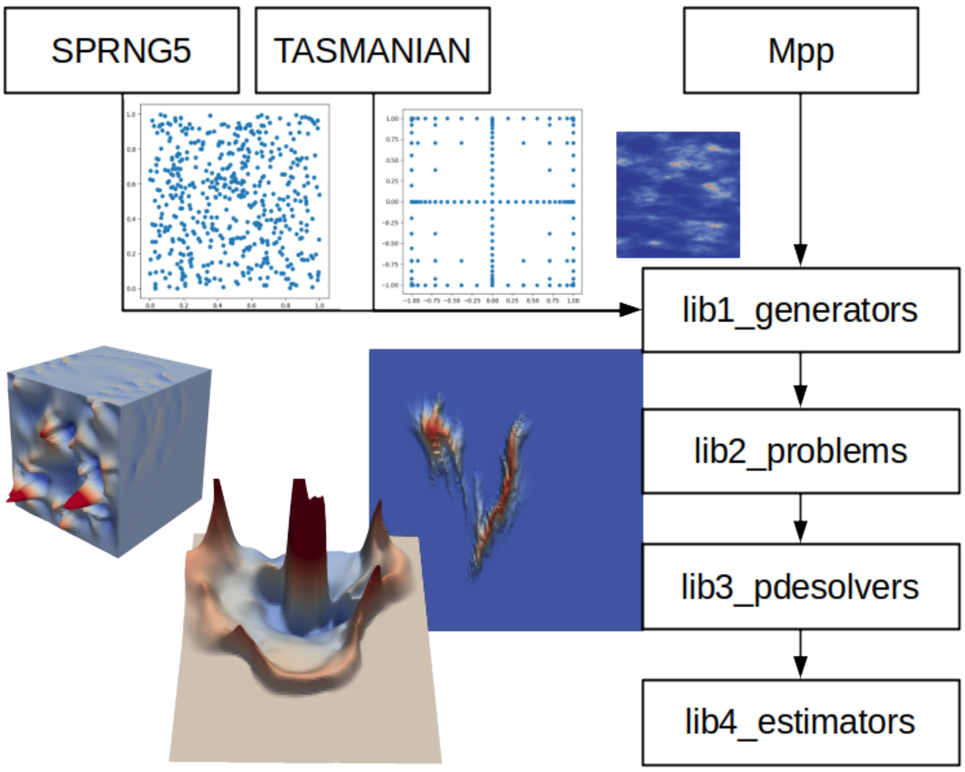


FIGURE 3.7: Software design of the UQ extension to M++.

f) The MIT Uncertainty Quantification Library (*MUQ*) [203] is a toolbox for solving forward and inverse uncertainty quantification problems. It is written in C++ but provides a Python interface. We refer to the website for example applications [194].

g) The software *XMC* [85] is developed within the ExaQUte project and provides implementations of MC, MLMC and CMLMC methods.

h) In [99] a generic implementation for Monte Carlo methods is introduced which can be used in combination with model solving software.

i) To overcome the issue of having to develop model solving software, i.e. FEM tools, as well as UQ software such as the tools above, *UM-Bridge* [251] was recently developed which bridges this gap by utilizing container technologies like Docker.

4 Numerical Results

In this chapter, we discuss the numerical results achieved with the proposed framework of this thesis. In particular, we demonstrate its capability to treat different PDEs subject to different uncertain input data. This includes the model problems 1.1-1.3 but is not restricted to them. In order to demonstrate the applicability, we make use of all presented discretizations of section 2.3 in combination with the BMLMC method of section 3.5.3. We investigate different problem configurations and algorithm combinations. All experiments are fully automated and can be repeated with the newest version of the software at any time. The presented results serve as a baseline which future developments on the method and the software have to verify or improve. The numerical results of each experiment can be summarized in several plots, each giving a different insight to the problem or the performance of the algorithm. We explain these plots using the standard elliptic PDE as in model problem 1.1 solved with Lagrange finite elements and the BMLMC method. We omit the details by referring to section 4.1 and focus instead on the illustration of the data in the following plots.

Numerical verification of assumptions (3.13) and (3.14). As pointed out in section 3.5, the assumptions 3.45 can be verified numerically on-the-fly. To this end, we present the measured asymptotic behavior of $\widehat{\mathrm{Y}}_\ell \approx \mathbb{E}[\mathrm{Y}_\ell] \coloneqq \mathbb{E}[\mathrm{Q}_\ell - \mathrm{Q}_{\ell-1}]$ and $s^2_{\mathrm{Y}_\ell} \approx \mathbb{V}[\mathrm{Y}_\ell] \coloneqq \mathbb{V}[\mathrm{Q}_\ell - \mathrm{Q}_{\ell-1}]$. This is represented by the dotted lines in the plots of figure 4.1 in a logarithmic scale. Furthermore, these plots verify that the development of $\widehat{\mathrm{Q}}_\ell \approx \mathbb{E}[\mathrm{Q}_\ell]$ and $s^2_{\mathrm{Q}_\ell} \approx \mathbb{V}[\mathrm{Q}_\ell]$ remains roughly constant over all levels and show the estimated exponents $\widehat{\alpha}$ and $\widehat{\beta}$ according to (3.25). These plots justify the application of multi-level estimators a posteriori and should always look similar to the ones presented here.

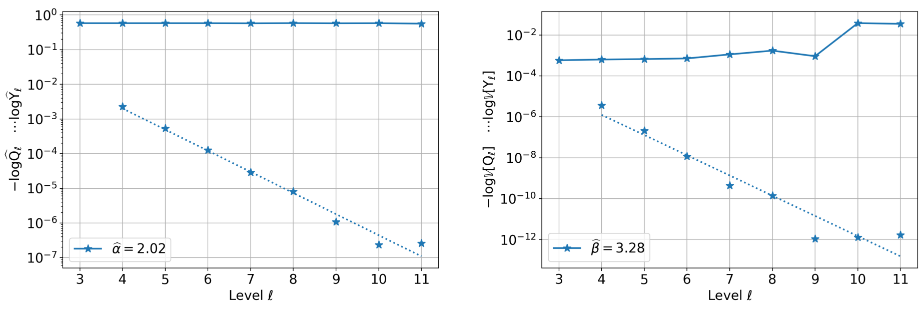


FIGURE 4.1: Numerical verification of assumptions (3.13) (left plot) and (3.14) (right plot).

Estimated optimal number of samples (3.23) and numerical verification of the constraint (3.29). In figure 4.2 on the left, the bar plot gives the number of samples $\{M_\ell\}_{\ell=0}^{L}$ drawn up until the last estimation round. This bar plot follows equation (3.22) and is computed via $\sum_{\mathtt{i}} \Delta M_{\mathtt{i},\ell}$ for each level $\ell = 0, \dots, L$. The plot on the right illustrates the cost distribution C_ℓ over the levels by a bar plot and the total cost of all levels as blue horizontal line. The total cost represented by the blue horizontal line should stay just below the red line representing the total cost budget B in #CPU · seconds. These two plots indicate that the BMLMC method was successfully executed and delivers feasible results.

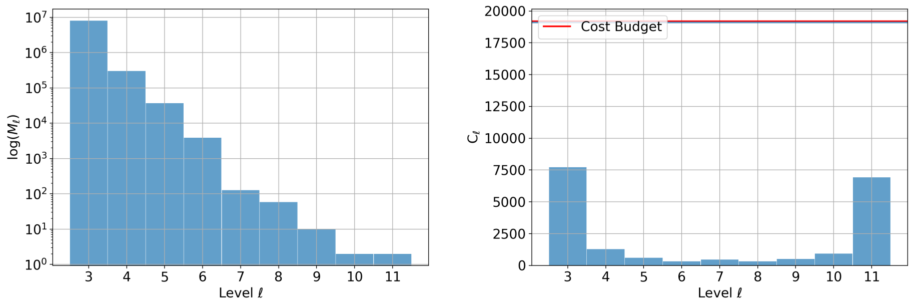


FIGURE 4.2: Total number of samples $\{M_\ell\}_{\ell=0}^{L}$ drawn following (3.22) (left plot). Cost distribution over the levels and verification of the budget restriction (3.29) (right plot).

Numerical verification of (3.27) and evaluation of the quantity of interest by (3.17) along with (3.30) as error estimation. The left plot of figure 4.3 numerically verifies the convergence given by equation (3.27), where the x-axis is the relative budget $\mathrm{C}_{\mathtt{i}}/\mathrm{B}_0$ and the y-axis is the estimated mean squared error $\widehat{\mathrm{err}}_{\mathrm{MSE}}$ (3.30) over the estimation rounds in logarithmic scales. On the right plot, the estimated quantity of interest $\widehat{\mathrm{Q}}_{\mathtt{i}}$ is given according to (3.17) over the relative budget $\mathrm{C}_{\mathtt{i}}/\mathrm{B}_0$ with error bars.

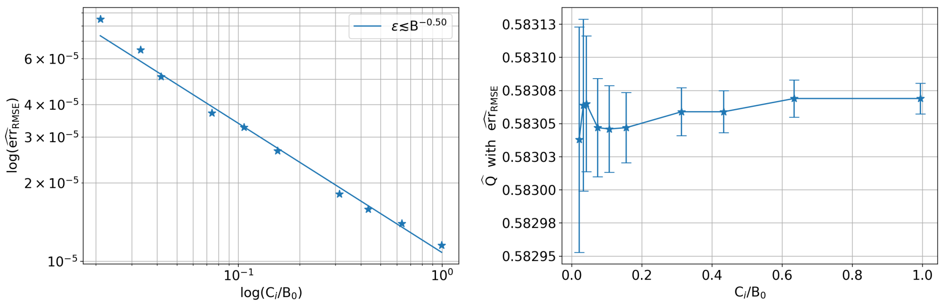


FIGURE 4.3: Numerical verification of convergence (3.27) (left plot) and evaluation of the quantity of interest by (3.17) along with (3.30) as error estimation (right plot).

Error development by (3.24) and (3.26) and verification of the assumption (3.15). Lastly, the left plot of figure 4.4 gives a more detailed insight into the algorithmic behavior. In particular, it shows the development of the discretization error estimated with (3.24) and the input error by (3.26) at different estimation rounds. It also shows the development of $\epsilon_{\mathtt{i}}$ acting as a frontier defining a target $\epsilon_{\mathtt{i}}$-box (dotted lines) the error minimization tries to reach each estimation round. Samples and levels are drawn until the total error estimation has crossed the current $\epsilon_{\mathtt{i}}$-box. Then, a new target RMSE is chosen. This plot further depicts the subsequent problem structure where each problem is associated with one $\epsilon_{\mathtt{i}}$-box. On the right plot of figure 4.4, the assumption (3.15) is numerically verified by the dotted line and the estimated $\widehat{\gamma}$ is shown. If this line does not follow the predicted cost growth over the levels, it can be a sign of badly parallelized computations. This usually stems from a poor choice of the initial sequence of samples $\left\{M_\ell^{\text{init}}\right\}_{\ell=0}^{L}$ leading to an inappropriate distribution of processes by equation (2.4).

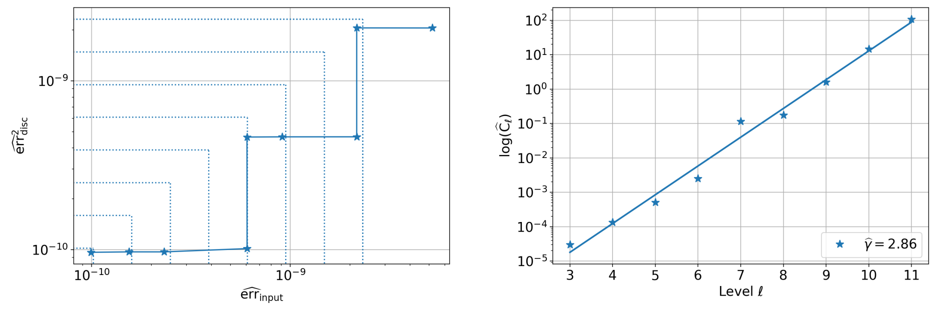


FIGURE 4.4: Development of the numerical error $\mathrm{err}_{\mathrm{disc}}$ and the input error $\mathrm{err}_{\mathrm{input}}$ alongside with RMSE tolerances $\epsilon_{\mathtt{i}}$ over different estimation rounds (left plot). Numerical verification (right plot) of assumption (3.15)

.

4.1 Experiments on the Elliptic Model Problem

We will now revisit model problem 1.1 and combine its properties described in 3.3.1 with the FEM convergence theory of 2.3.1 and the BMLMC method. In particular, we aim for convergence statements of the BMLMC method and verify them numerically. The presented theoretical results are based on [53, 246]. The theoretical predictions are verified by numerical results performed on the HoReKa supercomputer.

Model Problem 4.1. The weak discrete formulation to the model problem 1.1 considering the randomness in the input data reads as: Seek $u_\ell \in V_{\ell,\mathbf{p}\geq 1}^{\mathrm{Lag}}(u_{\mathrm{D}})$, such that

$$a_\omega(u_\ell(\omega,\cdot),\phi_\ell) = f_\omega(\phi_\ell), \quad \forall \phi_\ell \in V_{\ell,\mathbf{p}\geq 1}^{\mathrm{Lag}}(0)$$

for almost all $\omega \in \Omega$. We refer to 2.20 for the deterministic and discrete version of the problem and to 3.23 for the weak formulation with log-normal fields.

Theorem 4.2. *Let assumptions 3.22 be fulfilled and consider the case described in theorem 3.27 for some $0 < t \leq 1$. Then,*

$$\|(u-u_\ell)(\omega,\cdot)\|_{\mathrm{H}^1(D)} \lesssim \left(\frac{\kappa_{\max}(\omega)}{\kappa_{\min}(\omega)}\right)^{\frac{1}{2}} h_\ell^{\min\{s,\mathbf{p}\}} \, |u(\omega,\cdot)|_{\mathrm{H}^{\min\{s,\mathbf{p}\}+1}(D)}$$

for almost all but fixed $\omega \in \Omega$ and for all $0 < s < t$, such that corner singularities of the domain are controlled as well. Hence,

$$\|u-u_\ell\|_{\mathrm{L}^q(\Omega,\mathrm{H}^1(D))} \lesssim C_{\kappa,f} h_\ell^{\min\{s,\mathbf{p}\}}, \quad \|u-u_\ell\|_{\mathrm{L}^q(\Omega,\mathrm{L}^2(D))} \lesssim \tilde{C}_{\kappa,f} h_\ell^{\min\{2s,\mathbf{p}+1\}}$$

for all $q < q_*$ with the constants $C_{\kappa,f}$ and $\tilde{C}_{\kappa,f}$ only depending on the input data but not on h_ℓ.

Proof. Follows by proposition 2.21 and lemma 3.24. The arguments are given in [53, 246]. $\square$

We will now combine this result with the assumptions 3.45 to deduce convergence of the BMLMC method.

Proposition 4.3. *Suppose that D is a polygonal domain and that assumptions 3.22 hold for some $0 < t < 1$ and $1 \leq q < \frac{q^*}{2}$. For $\mathrm{Q}(u(\omega)) = \|u\|_{\mathrm{H}^1(D)}^q$ the assumptions 3.45 a) and b) hold for any $\alpha < t$ and $\beta < 2t$. For $t = 1$ we get $\alpha = 1$ and $\beta = 2$. Likewise, for $\mathrm{Q}(u(\omega)) = \|u\|_{\mathrm{L}^2(D)}^q$ the assumptions 3.45 a) and b) hold for any $\alpha < 2t$ and $\beta < 4t$. For $t = 1$ we get $\alpha = 2$ and $\beta = 4$.*

Proof. Follows by theorem 4.2. We refer to [53, proposition 4.2 and 4.3] for details. $\square$

Having the above, we conclude the convergence of RMSE with respect to the computational budget in table 4.1 for the $|\cdot|_{\mathrm{H}^1(D)}$- and $\|\cdot\|_{\mathrm{L}^2(D)}$-norm as QoI. The results in this table consider log-normal fields with a Gaussian covariance kernel (3.1) (left table $\nu = 1$) and an exponential covariance kernel (right table $\nu = 2$).

Remark 4.4. Other functionals than the L^2- or the H^1-norm can be considered as well. In [246, section 3] it is shown that convergence for further linear and nonlinear functionals can be derived by using a duality argument.

d	$\lvert\cdot\rvert_{\mathrm{H}^1(D)}$	$\lVert\cdot\rVert_{\mathrm{L}^2(D)}$
1	$\mathrm{B}^{-1/2}$	$\mathrm{B}^{-1/2}$
2	$\mathrm{B}^{-1/2}$	$\mathrm{B}^{-1/2}$
3	$\mathrm{B}^{-1/3}$	$\mathrm{B}^{-1/2}$

d	$\lvert\cdot\rvert_{\mathrm{H}^1(D)}$	$\lVert\cdot\rVert_{\mathrm{L}^2(D)}$
1	$\mathrm{B}^{-1/2}$	$\mathrm{B}^{-1/2}$
2	$\mathrm{B}^{-1/4}$	$\mathrm{B}^{-1/2}$
3	$\mathrm{B}^{-1/6}$	$\mathrm{B}^{-1/3}$

Table 4.1: The left table is for a covariance kernel with $\nu = 2$. The right table is for a covariance kernel with $\nu = 1$. The table was taken from [53] and translated to the budgeted method.

Thereby, the first takeaway from this section is that the smoothness of the random field directly transmits into the convergence of the BMLMC method. In the following, we will separate the experiments in two classes. The first class consists of model problem experiments where we investigate the influence of the input data on the convergence and behavior of the method. By doing so, we can numerically study the regularity of the problem. In the second class of experiments we investigate the methodology, i.e., everything from the parallel data structure to the finite element discretization and the hyperparameters of the BMLMC method. Unless stated otherwise, we consider the following problem and model configuration within this section.

Problem Configuration 4.5. We search for the solution $u\colon \Omega \times [0,1]^2 \to \mathbb{R}$, such that

$$\begin{cases} -\operatorname{div}(\kappa(\omega,\mathbf{x})\nabla u(\omega,\mathbf{x})) &= 0 & \mathbf{x} \in (0,1)^2 \\ \kappa(\omega,\mathbf{x})\nabla u(\omega,\mathbf{x})\cdot\mathbf{n} &= -1 & \mathbf{x} \in \Gamma_{\mathrm{N}}^1 = \{\mathbf{x} \in \partial D\colon x_2 = 1\} \\ \kappa(\omega,\mathbf{x})\nabla u(\omega,\mathbf{x})\cdot\mathbf{n} &= 0 & \mathbf{x} \in \Gamma_{\mathrm{N}}^2 = \{\mathbf{x} \in \partial D\colon x_1 \in \{0,1\}\} \\ u(\omega,\mathbf{x}) &= 0 & \mathbf{x} \in \Gamma_{\mathrm{D}} = \{\mathbf{x} \in \partial D\colon x_2 = 0\} \end{cases},$$

where the permeability $\kappa\colon \Omega \times D \to \mathbb{R}$ is log-normally distributed with covariance function (3.6) for which we choose $\sigma = 1.0$, $\boldsymbol{\lambda} = (0.15, 0.15)^\top$ and $\nu = 1.0$ as default parameters. The default QoI is the L^2-norm over the whole domain.

Method Configuration 4.6. We search for numerical solutions in $V_{\ell,\mathbf{p}=1}^{\mathrm{Lag}}$ and initialize the BMLMC method with $\{M_{0,\ell}\}_{\ell=0}^{L} = \{M_{0,3} = 2^{14},\, M_{0,4} = 2^{13},\, M_{0,5} = 2^{11},\, M_{0,6} = 2^{10},\, M_{0,7} = 2^{8}\}$ which consumes less than two percent of the total computational budget. The computation is deployed on the HoReKa supercomputer for 10 minutes with $|\mathcal{P}| = 32$ processes on a single node. The hyperparameters of the method are chosen as $\theta = 0.5$ and $\eta = 0.9$ and the arising linear system is solved with the GMRES method and an incomplete LU-decomposition as preconditioner. We refer to {40} for the complete configuration of all upcoming jobs.

Problem Experiments 4.7. We investigate the role of the covariance function (3.6) as in [28] in further detail. First, we examine the influence of the correlation lengths $\boldsymbol{\lambda}$ in figure 4.5. This is of particular interest since the finite element discretization should resolve the smallest structure of the permeability (confer remark 3.47). The lowest mesh resolution in this particular experiment is $h_0 = 0.25$, and thus the correlation lengths $\boldsymbol{\lambda} \in \{(0.05, 0.05)^\top, (0.1, 0.1)^\top, (0.15, 0.15)^\top\}$ undercut h_0. On the bar plot on the left of figure 4.5 we observe the following: the shorter the correlation length, the fewer samples are drawn on the coarse and the more samples are taken on the finer grids. The opposite behavior can be seen for the longer correlation lengths. This effect transmits to the cost distribution depicted on the right, i.e., the simulation of the problem with the longer correlation lengths spend a larger amount of the computational budget on the lower levels and less amount of the budget on the higher levels. Hence, the algorithm adapts to the problem without a priori knowledge. By doing so, we also might be able to deduce properties about the problem by the behavioral quantities of the algorithm like the drawn sample amount on each level.

Next, we discuss the influence of the smoothing parameter ν by choosing $\nu \in \{1.0, 1.4, 1.8\}$ for the covariance function. Such as in table 4.1, we consider the L^2-norm in figure 4.6 and the H^1-norm in figure 4.7. First, we observe that the multi-level estimator is applicable to this type of problem by the first row of each plot. Furthermore, as predicted in proposition 4.3 the rates $\widehat{\alpha}$ and $\widehat{\beta}$ grow as ν grows in the H^1-norm. This is due to the increased Hölder continuity of the samples given by the estimate in theorem 3.27. This also matches results reported in [246, 53] and stated in table 4.1. We further remark, that we had to increase from $|\mathcal{P}| = 32$ to $|\mathcal{P}| = 128$ processing units for the H^1-experiment due to the fact that the convergence rate in the H^1 is lower and thereby more levels are drawn during the ten minutes of computation. The resources are added to cope with the higher memory demand of the finer grids by distributing the mesh on more processing units.

The last experiment on the random field considers a varying $\sigma \in \{1.0, 1.5, 2.0\}$ in the covariance function (3.6). By increasing σ, the amplitudes of the random field get stretched. Hence, the constant in (3.27) increases with an increasing σ. This can be nicely observed in figure 4.8. For this last experiment on the random field, we also present the evaluation of the QoI as well as the load distribution over the levels. We see in figure 4.8 that the cost is spread over all levels and that we stay within the cost budget for all simulations. We lastly draw the attention to the evaluation of the QoI in the lower right plot, where we clearly see that we solve different problems with different solutions.

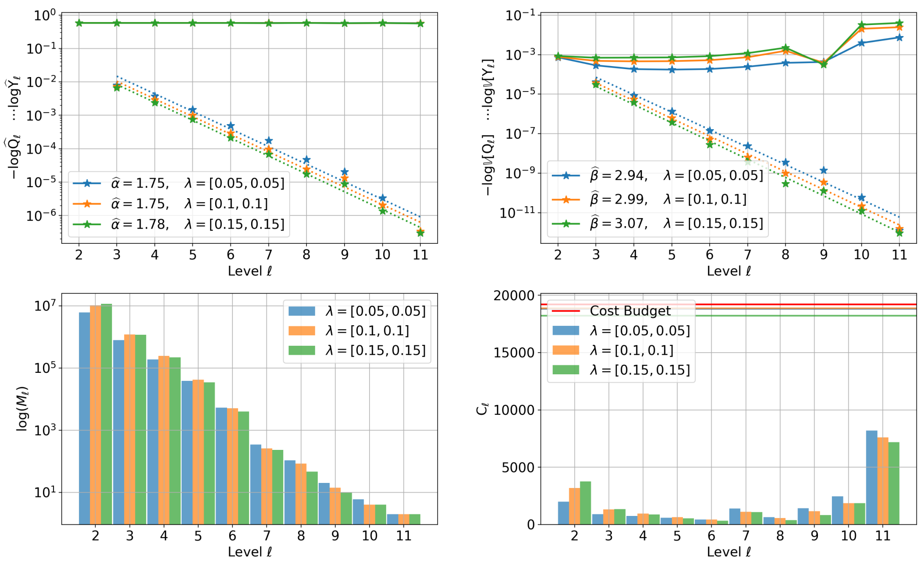


FIGURE 4.5: BMLMC method with configuration 4.6 applied to the elliptic problem configuration 4.5 with varying correlation length of the covariance function.

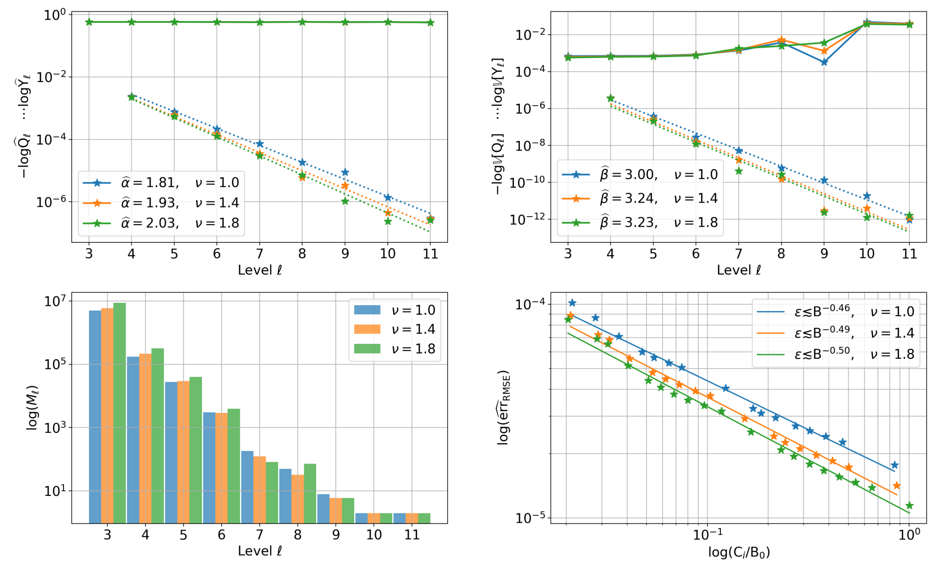


Figure 4.6: BMLMC method with configuration 4.6 applied to the elliptic problem configuration 4.5 with varying smoothness of the covariance function and the L^2-norm as QoI.

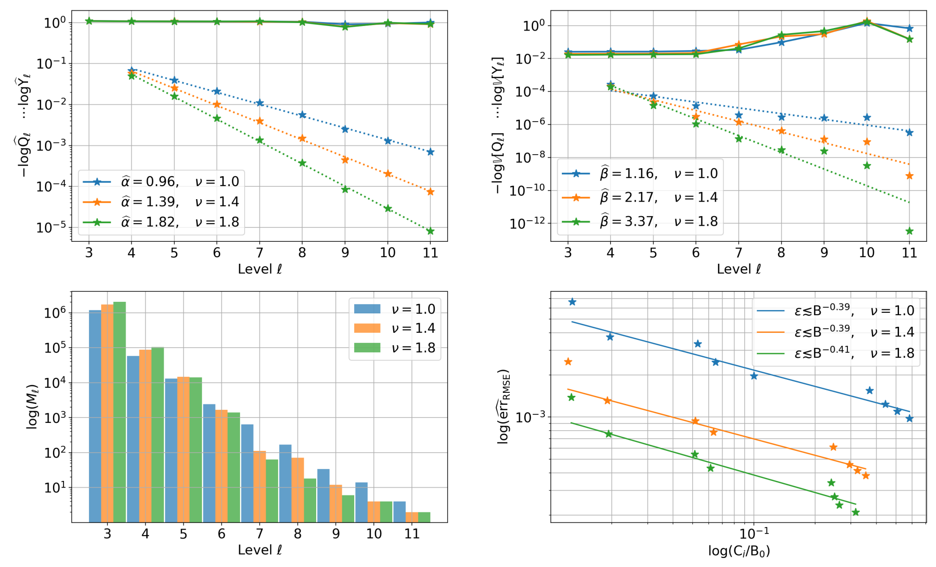


Figure 4.7: BMLMC method with configuration 4.6 applied to the elliptic problem configuration 4.5 with varying smoothness of the covariance function and the H^1-norm as QoI.

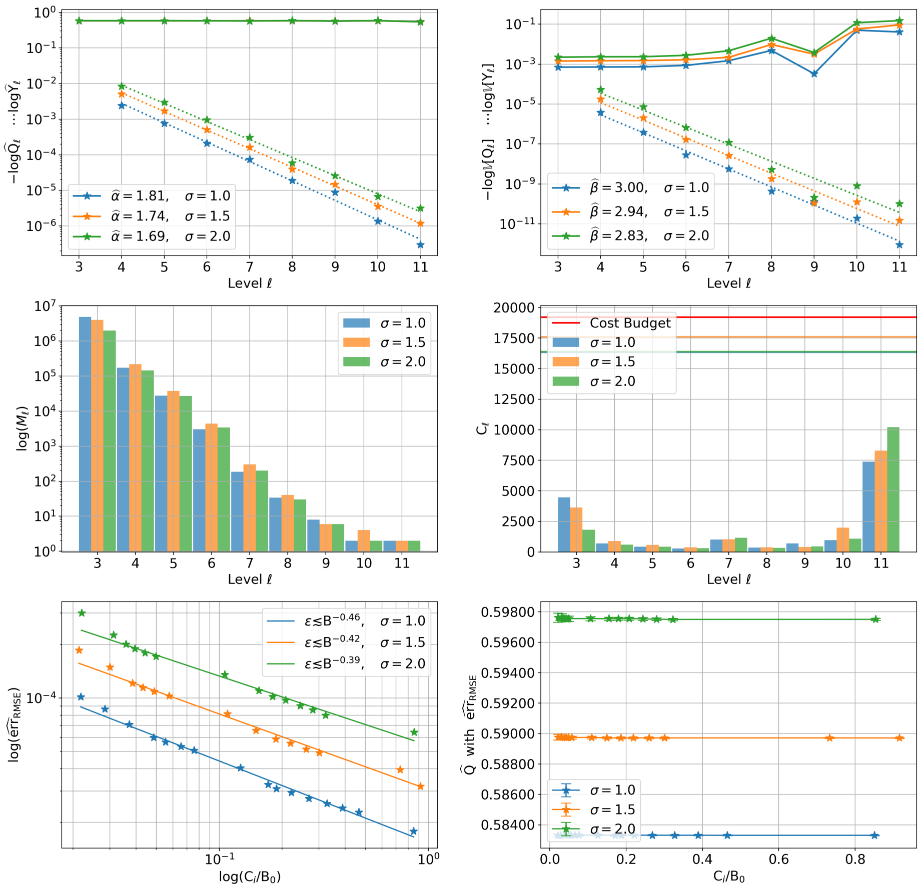


FIGURE 4.8: BMLMC method with configuration 4.6 applied to the elliptic problem configuration 4.5 with varying variance of the covariance function.

Method Experiments 4.8. We switch the focus from the problem to the discretization. As a start, we consider the problem in default configuration 4.5 and increase the polynomial degree of the Lagrange discretization. By theorem 4.2 we do not expect a faster convergence since the default problem simply does not supply enough regularity to leverage the higher degree. In fact, we can see in figure 4.9 in the lower right plot, that a larger polynomial degree even yields a larger estimated RMSE. This is explained by the increased computational cost of the higher degree leading to larger errors, if the same computational budget was utilized.

In section 2.3, we further explored the application of different discretizations to the elliptic model problem. The intent of the following experiment is to figure out which discretization for the default problem 4.5 yields the smallest error estimation. By the propositions 2.21, 2.24, 2.26 and lemma 3.24, we do expect the same convergence rates for all discretizations. However, the computational demand of these discretizations is different. Figure 4.10 depicts the expected results of the experiment, i.e., the same convergence rates but a larger estimated RMSEs, if the discretization approach is more expensive. A rather surprising result is that the hybridized mixed discretization performs worse than the standard mixed approach. We suspect that the reason for this is the sample parallelism and that the advantage of the hybridized model of cell-wise independent computations can not be exploited anymore. However, searching the solution with respect to another norm might result in a completely different outcome of the experiment. We leave the investigation of other quantities of interest open for further experiments.

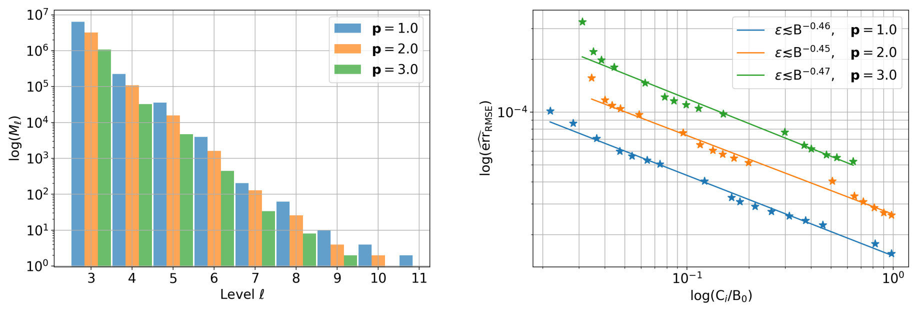


FIGURE 4.9: BMLMC method with configuration 4.6 applied to the elliptic problem configuration 4.5 with varying polynomial degree **p** of the Lagrange finite element space.

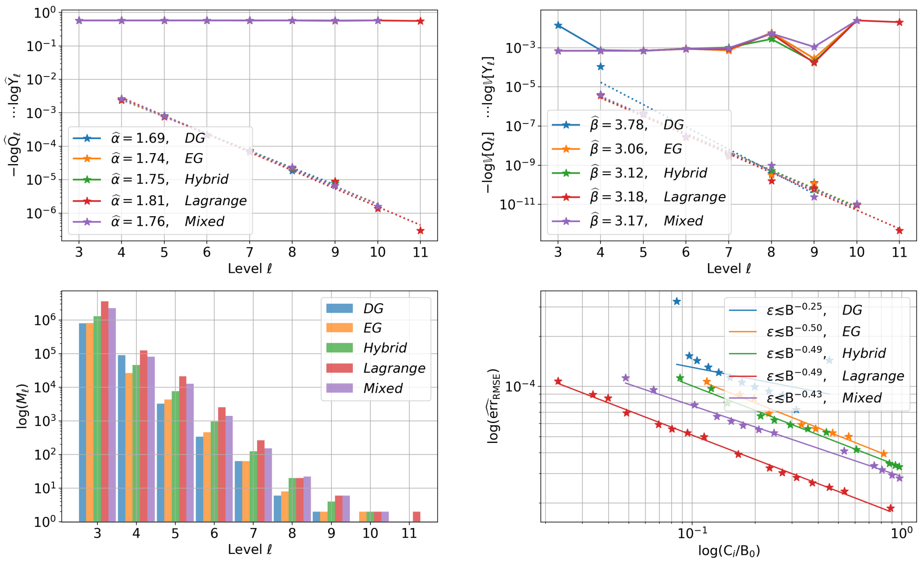


FIGURE 4.10: BMLMC method with configuration 4.6 applied to the elliptic problem configuration 4.5 employing all introduced discretizations of section 2.3.1.

Method Experiments 4.9. As a central piece of this thesis, we continue our experiments by investigating the parallelism proposed in section 2.1.3 and 2.4.3. As a start, we compare the multi-mesh parallelization feature (*On*) with the naive parallelization only on the domain (*Off*) in figure 4.11. The lower left plot verifies that both methods indeed compute the same QoI. However, the error admitted by the fully parallelized method is significantly smaller even though both methods were executed with the same computational resources, i.e., with $|\mathcal{P}| = 32$ processing units and $\mathrm{T} = 600$ seconds of computing time. The error reduction by utilizing the resources in a smarter way can also be observed in the lower left plot. By recalling the convergence theorem of the BMLMC method 3.54, we recognise that the hidden constant is heavily dependent on the parallelization approach. Lastly, the plots in the top row of figure 4.11 show that the naively parallelized method draws fewer samples on all levels and that the bottleneck for the cost is the lowest level. This stands in contrast to the fully parallelized method which distributes the computational load nicely over all levels resulting in much better overall performance. Similar experiments for other PDEs will follow in the upcoming sections, however, the discussion remains the same and only significantly more resources will be allocated.

Next, we pick up on the discussion at the end of section 3.5, i.e., we investigate the strong and the weak scaling of the parallelization. The strong scaling experiment is given in figure 4.12. Here, we have assigned each experiment the same computational budget of $\mathrm{B} = |\mathcal{P}| \cdot \mathrm{T} = 38400\ \mathrm{CPU} \cdot \mathrm{seconds}$ while compensating computing time with processing units. In particular, we divided the initial 40 minutes of computing time three times by two and sequentially doubled the amount of process from $|\mathcal{P}| = 16$ up to $|\mathcal{P}| = 128$. We recall, if the algorithm does scale strongly and the serial part is very small, the parallelization should deliver very similar results for all configurations by Amdahl's law (3.33). Indeed, all methods yield the same QoI, however, in this figure we can also observe the limits of the parallelization approach. In the case of $|\mathcal{P}| = 128$, we seem to approach the limits of Amdahl's law since we are not capable to exhaust the budget completely and the sequential parts of the algorithm become more dominant. On the other hand, for $|\mathcal{P}| = 16$ we can not exhaust the budget completely either, because one individual processing unit has a large computational load on the higher levels. The assigned subdomain to one process becomes huge such that the memory access of this process becomes a bottleneck or even leads to a memory overflow. By drawing the attention one more time to the lower left plot, we can also detect the in section 3.5 predicted performance losses due to communication and idling processes. More workers require more communication which directly transmits to a worse constant in the convergence theorem 3.54. Thereby, the goal of further optimizations of the parallelism should be to reduce communication and idling processes. Concluded, we observe strong scaling only in a window $[|\mathcal{P}|_{\min}, |\mathcal{P}|_{\max}]$ of processes which is similar to the parallelism of a standard FEM solver.

In the last parallelization experiment we investigate the weak scaling behavior. As both parts of the total parallelization approach scale weakly, i.e., the FEM solver scales weakly and the Monte Carlo method scales weakly, we expect the BMLMC method with the MSS parallelization to scale weakly as well. To this end, we fix the computing time on ten minutes and sequentially double amount of processing units, starting from $|\mathcal{P}| = 16$ up to $|\mathcal{P}| = 128$. We then follow the remark to equation (3.34) and measure the error reduction rate by utilizing more processing units via

$$\frac{\mathrm{err}_{\mathrm{RMSE}}^{|P|=128}}{\mathrm{err}_{\mathrm{RMSE}}^{|P|=16}} = 0.5261 \overset{!}{=} 8^{e} \quad \Leftrightarrow \quad e = -0.31.$$

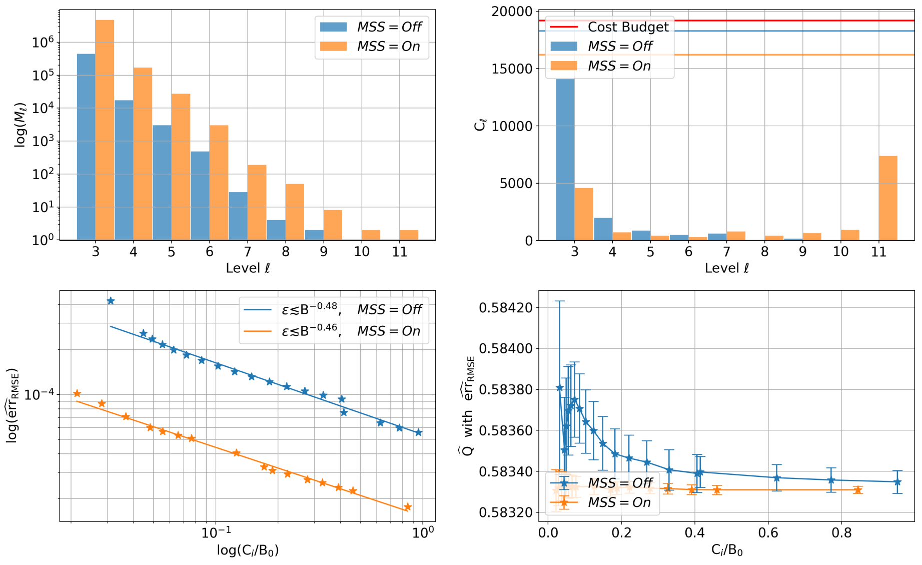


Figure 4.11: BMLMC method with configuration 4.6 applied to the elliptic problem configuration 4.5. Comparison of MSS parallelism with naive parallelization.

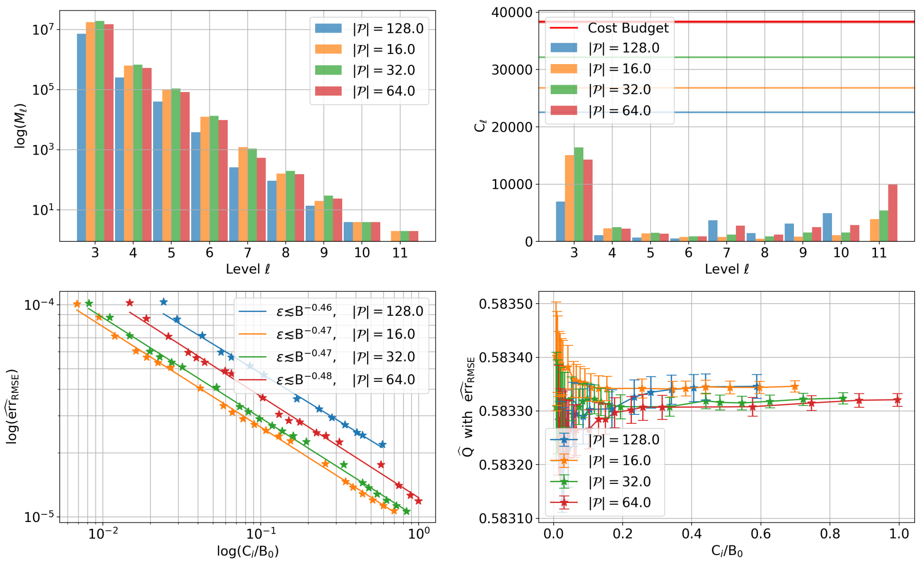


Figure 4.12: BMLMC method with configuration 4.6 applied to the elliptic problem configuration 4.5 while measuring the strong scaling of the parallelism.

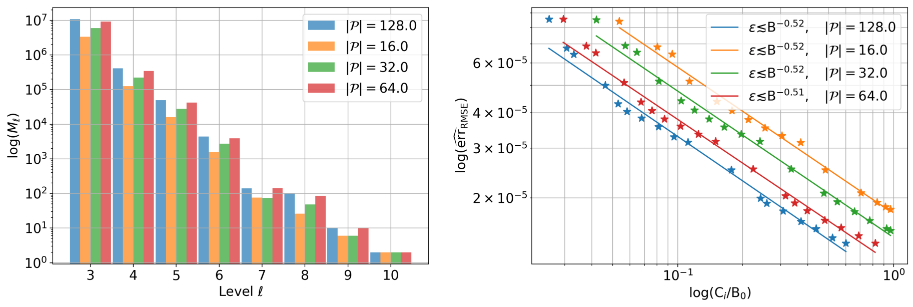


FIGURE 4.13: BMLMC method with configuration 4.6 applied to the elliptic problem configuration 4.5 while measuring the weak scaling of the parallelism.

Method Experiments 4.10. So far, we have not argued for nor against a choice of η and θ. We present two experiments with varying hyperparameters θ and η to settle on a parameter choice. First, we considered three different splitting parameters $\theta \in \{0.3, 0.5, 0.7\}$ and applied the default method 4.6 to the default problem 4.5. The results are shown in figure 4.14. We recall that θ is responsible for splitting the MSE, i.e.,

$$(\mathbb{E}[\mathrm{Q}_L - \mathrm{Q}])^2 < (1-\theta)\epsilon^2 \quad \text{and} \quad \mathbb{V}[\widehat{\mathrm{Q}}^{\mathrm{MLMC}}_{\{M_\ell\}_{\ell=0}^L}] < \theta\epsilon^2.$$

This results in solving subproblems of different structure since θ determines the length and the width of the target $\epsilon_{\mathtt{i}}$-boxes. This can be seen in figure 4.14 on the left. In particular, for $\theta = 0.3$ the variance reduction, and for $\theta = 0.7$ the bias reduction is favored by the method. For this experiment, we conclude that $\theta = 0.5$ is the best choice. However, we further remark that we have tried an adaptive scheme to the relative budget $\mathrm{C}_{\mathtt{i}}/\mathrm{B}_0$

$$\theta_{\mathtt{i}} = 0.7 \cdot (1 - \mathrm{C}_{\mathtt{i}}/\mathrm{B}_0) + 0.3 \cdot \mathrm{C}_{\mathtt{i}}/\mathrm{B}_0$$

in the upcoming model problems. The idea is that the method draws new levels more likely at the beginning of the simulation to improve the reliability. While this brings the advantage of exploring the levels faster, the subproblems do not possess a nested structure anymore. Hence, this technique sacrifices the optimality condition such that we choose $\theta = 0.5$ for all experiments.

Next, we consider the parameter η. The results are shown in figure 4.15 for $\eta \in \{0.7, 0.8, 0.9\}$. Prior to the experiment, we expected that this might have an influence on the load distributions since $\Delta M_{\mathtt{i},\ell}$ is heavily dependent on η. However, by figure 4.15 we can conclude that taking larger steps in the minimization via a smaller η does not bring any benefits for the load distribution and therefore $\eta = 0.9$ seems to be a reasonable value yielding many estimation points without compromising on the load distribution.

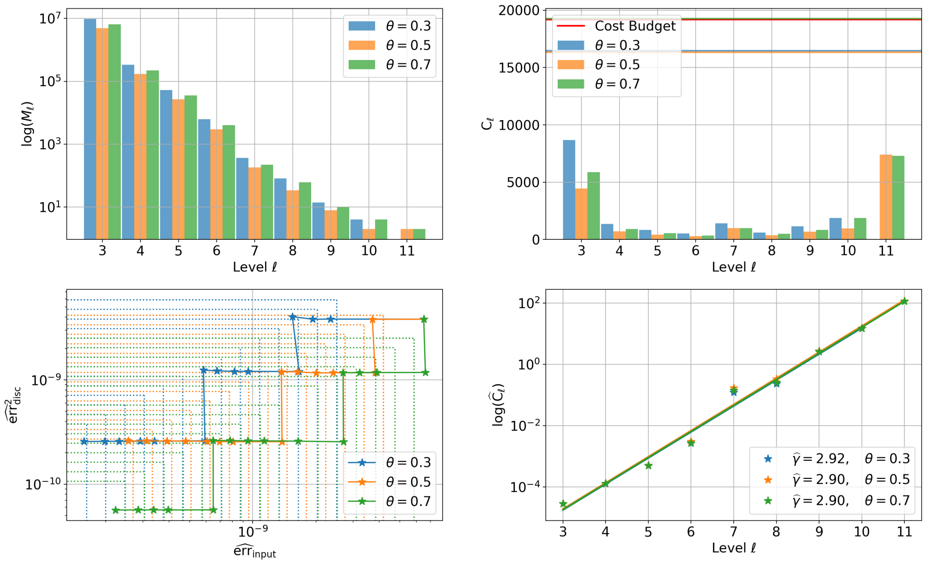


Figure 4.14: BMLMC method with configuration 4.6 applied to the elliptic problem configuration 4.5 while using different splitting parameters $\theta \in \{0.3, 0.5, 0.7\}$.

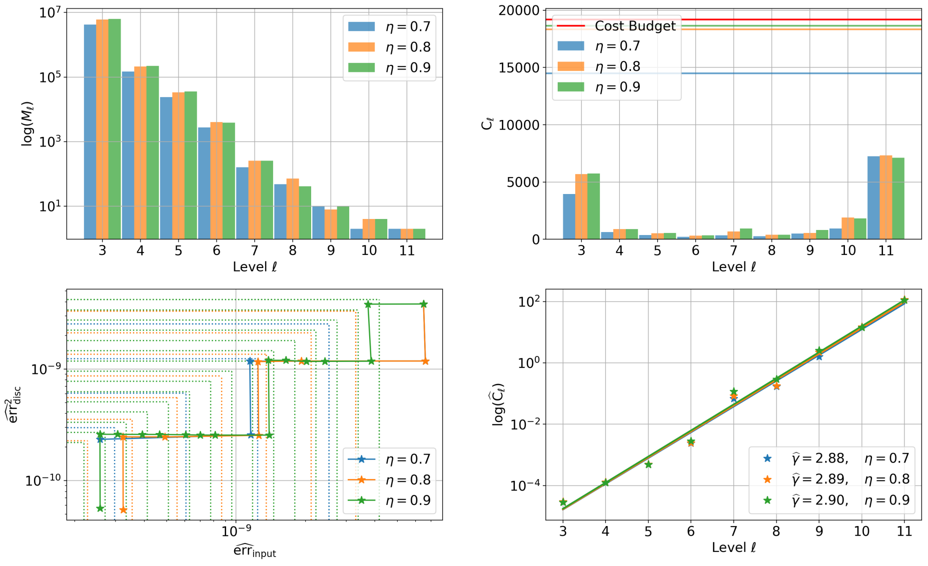


Figure 4.15: BMLMC method with configuration 4.6 applied to the elliptic problem configuration 4.5 while using different ϵ-reduction parameters $\eta \in \{0.7, 0.8, 0.9\}$.

Method Experiments 4.11. Lastly for the elliptic model problem, we shortly discuss figure 4.16 where we applied different preconditioners to the default problem 4.5. In particular, we used a simple Jacobi and Gauss-Seidel preconditioner which perform worse than the incomplete *LU* decomposition and also worse than a multigrid preconditioner. However, by this figure we can deduce that the multigrid preconditioner gives the best results. In particular, a detailed analysis of the computational data has shown that as soon as higher levels get involved in the simulation the multigrid approach is the best preconditioner choice. The amount of pre- and postsmoothing steps had no big influence on the results.

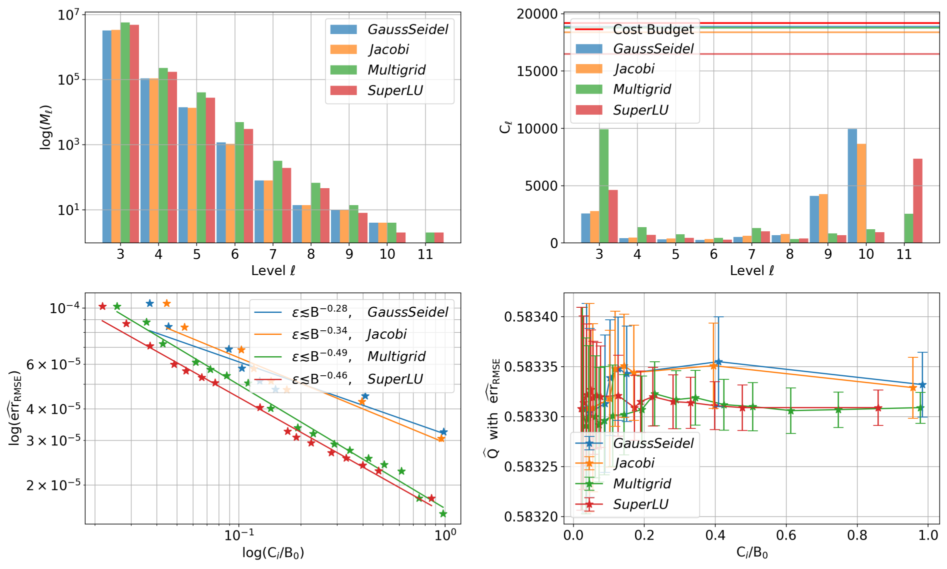


FIGURE 4.16: BMLMC method with configuration 4.6 applied to the elliptic problem configuration 4.5 while employing different preconditioners for the multi-sample system.

Problem	**Parameter Choice**	**Method**	**Parameter Choice**	$(\widehat{\alpha}, \widehat{\beta}, \widehat{\gamma})$	**Convergence**	**RMSE**	**Figure**
Elliptic Configuration 4.5	$\lambda = (0.05, 0.05)$	Configuration 4.6	default	$(1.75, 2.94, 2.79)$	$\mathrm{B}^{-0.44}$	$2.53 \cdot 10^{-5}$	4.5
	$\lambda = (0.10, 0.10)$			$(1.75, 2.99, 2.81)$	$\mathrm{B}^{-0.44}$	$1.93 \cdot 10^{-5}$	
	$\lambda = (0.15, 0.15)$			$(1.78, 3.07, 2.80)$	$\mathrm{B}^{-0.45}$	$1.71 \cdot 10^{-5}$	
Elliptic Configuration 4.5	$\nu = 1.0,\ \lVert\cdot\rVert_{\mathrm{L}^2(D)}$	Configuration 4.6	default	$(1.81, 3.00, 2.88)$	$\mathrm{B}^{-0.46}$	$1.77 \cdot 10^{-5}$	4.6
	$\nu = 1.4,\ \lVert\cdot\rVert_{\mathrm{L}^2(D)}$			$(1.93, 3.24, 2.87)$	$\mathrm{B}^{-0.49}$	$1.42 \cdot 10^{-5}$	
	$\nu = 1.8,\ \lVert\cdot\rVert_{\mathrm{L}^2(D)}$			$(2.03, 3.23, 2.86)$	$\mathrm{B}^{-0.50}$	$1.14 \cdot 10^{-5}$	
Elliptic Configuration 4.5	$\nu = 1.0,\ \lVert\cdot\rVert_{\mathrm{H}^1(D)}$	Configuration 4.6	$\lvert\mathcal{P}\rvert = 128$	$(0.96, 1.16, 2.97)$	$\mathrm{B}^{-0.39}$	$9.79 \cdot 10^{-4}$	4.7
	$\nu = 1.4,\ \lVert\cdot\rVert_{\mathrm{H}^1(D)}$			$(1.39, 2.17, 3.05)$	$\mathrm{B}^{-0.39}$	$3.83 \cdot 10^{-4}$	
	$\nu = 1.8,\ \lVert\cdot\rVert_{\mathrm{H}^1(D)}$			$(1.82, 3.37, 3.08)$	$\mathrm{B}^{-0.41}$	$2.10 \cdot 10^{-4}$	
Elliptic Configuration 4.5	$\sigma = 1.0$	Configuration 4.6	default	$(1.81, 3.00, 2.89)$	$\mathrm{B}^{-0.46}$	$1.78 \cdot 10^{-5}$	4.8
	$\sigma = 1.5$			$(1.74, 2.94, 2.91)$	$\mathrm{B}^{-0.42}$	$3.18 \cdot 10^{-5}$	
	$\sigma = 2.0$			$(1.69, 2.83, 2.94)$	$\mathrm{B}^{-0.39}$	$6.43 \cdot 10^{-5}$	
Elliptic Configuration 4.5	default	Configuration 4.6	$\mathbf{p} = 1$	$(1.81, 3.03, 2.88)$	$\mathrm{B}^{-0.46}$	$1.56 \cdot 10^{-5}$	4.9
			$\mathbf{p} = 2$	$(1.73, 2.91, 3.12)$	$\mathrm{B}^{-0.45}$	$2.59 \cdot 10^{-5}$	
			$\mathbf{p} = 3$	$(1.61, 2.89, 3.28)$	$\mathrm{B}^{-0.47}$	$5.24 \cdot 10^{-5}$	
Elliptic Configuration 4.5	default	Configuration 4.6	DG	$(1.69, 3.78, 2.82)$	$\mathrm{B}^{-0.25}$	$1.44 \cdot 10^{-4}$	4.10
			EG	$(1.74, 3.06, 2.68)$	$\mathrm{B}^{-0.50}$	$3.95 \cdot 10^{-5}$	
			Hybrid	$(1.75, 3.12, 2.62)$	$\mathrm{B}^{-0.49}$	$3.30 \cdot 10^{-5}$	
			Lagrange	$(1.81, 3.18, 2.78)$	$\mathrm{B}^{-0.49}$	$1.87 \cdot 10^{-5}$	
			Mixed	$(1.76, 3.17, 3.04)$	$\mathrm{B}^{-0.43}$	$2.82 \cdot 10^{-5}$	
Elliptic Configuration 4.5	default	Configuration 4.6	MSS Off	$(1.71, 4.48, 2.00)$	$\mathrm{B}^{-0.48}$	$5.58 \cdot 10^{-5}$	4.11
			MSS On	$(1.81, 2.99, 2.89)$	$\mathrm{B}^{-0.46}$	$1.77 \cdot 10^{-5}$	
Elliptic Configuration 4.5	default	Configuration 4.6	$\lvert\mathcal{P}\rvert = 128,\ \mathrm{T} = 5\mathrm{min}$	$(1.75, 2.60, 3.16)$	$\mathrm{B}^{-0.46}$	$2.20 \cdot 10^{-5}$	4.12
			$\lvert\mathcal{P}\rvert = 64,\ \mathrm{T} = 10\mathrm{min}$	$(1.82, 2.88, 2.97)$	$\mathrm{B}^{-0.48}$	$1.19 \cdot 10^{-5}$	
			$\lvert\mathcal{P}\rvert = 32,\ \mathrm{T} = 20\mathrm{min}$	$(1.82, 3.05, 2.87)$	$\mathrm{B}^{-0.47}$	$1.06 \cdot 10^{-5}$	
			$\lvert\mathcal{P}\rvert = 16,\ \mathrm{T} = 40\mathrm{min}$	$(1.82, 3.23, 2.79)$	$\mathrm{B}^{-0.47}$	$1.07 \cdot 10^{-5}$	
Elliptic Configuration 4.5	default	Configuration 4.6	$\lvert\mathcal{P}\rvert = 128,\ \mathrm{T} = 10\mathrm{min}$	$(2.21, 3.39, 3.10)$	$\mathrm{B}^{-0.52}$	$1.38 \cdot 10^{-5}$	4.13
			$\lvert\mathcal{P}\rvert = 64,\ \mathrm{T} = 10\mathrm{min}$	$(2.20, 3.57, 2.96)$	$\mathrm{B}^{-0.51}$	$1.38 \cdot 10^{-5}$	
			$\lvert\mathcal{P}\rvert = 32,\ \mathrm{T} = 10\mathrm{min}$	$(2.21, 3.66, 2.88)$	$\mathrm{B}^{-0.52}$	$1.54 \cdot 10^{-5}$	
			$\lvert\mathcal{P}\rvert = 16,\ \mathrm{T} = 10\mathrm{min}$	$(2.21, 3.85, 2.77)$	$\mathrm{B}^{-0.52}$	$1.83 \cdot 10^{-5}$	
Elliptic Configuration 4.5	default	Configuration 4.6	$\theta = 0.3$	$(1.74, 2.98, 2.92)$	$\mathrm{B}^{-0.49}$	$1.95 \cdot 10^{-5}$	4.14
			$\theta = 0.5$	$(1.81, 3.00, 2.90)$	$\mathrm{B}^{-0.46}$	$1.77 \cdot 10^{-5}$	
			$\theta = 0.7$	$(1.81, 3.03, 2.90)$	$\mathrm{B}^{-0.43}$	$1.56 \cdot 10^{-5}$	
Elliptic Configuration 4.5	default	Configuration 4.6	$\eta = 0.7$	$(1.83, 3.03, 2.88)$	$\mathrm{B}^{-0.51}$	$1.76 \cdot 10^{-5}$	4.15
			$\eta = 0.8$	$(1.82, 3.05, 2.89)$	$\mathrm{B}^{-0.48}$	$1.54 \cdot 10^{-5}$	
			$\eta = 0.9$	$(1.81, 3.03, 2.90)$	$\mathrm{B}^{-0.46}$	$1.57 \cdot 10^{-5}$	
Elliptic Configuration 4.5	default	Configuration 4.6	Gauss-Seidel	$(1.44, 2.43, 3.35)$	$\mathrm{B}^{-0.28}$	$3.25 \cdot 10^{-5}$	4.16
			Jacobi	$(1.53, 2.47, 3.28)$	$\mathrm{B}^{-0.34}$	$3.06 \cdot 10^{-5}$	
			Multigrid	$(1.82, 3.05, 2.65)$	$\mathrm{B}^{-0.49}$	$1.55 \cdot 10^{-5}$	
			SuperLU	$(1.81, 3.00, 2.89)$	$\mathrm{B}^{-0.46}$	$1.78 \cdot 10^{-5}$	

Table 4.2: Overview on all numerical results considering the elliptic model problem.

4.2 Experiments on the Transport Model Problem

We now move to a computationally more challenging problem, namely model problem 1.2. We approximate the randomized model problem in 3.3.3 with the methods of section 2.3.3 and conduct very similar experiments as already presented, however, the computational load is much larger. We used the HoReKa supercomputer and considered the following weak, semi-discrete and randomized model problem.

Model Problem 4.12. The semi-discrete formulation to the model problem 1.2 considering the randomness in the input data as described in model 3.34 reads as: Seek for almost all fixed $\omega \in \Omega$ the solution $u_\ell \in V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0}$ such that

$$\langle \underline{M}_\ell \partial_t u_\ell(\omega), \phi_\ell \rangle_{\mathrm{L}^2(D)} + \langle \underline{A}_\ell(\omega) u_\ell(\omega), \phi_\ell \rangle_{\mathrm{L}^2(D)} = 0 \quad \text{for} \quad t \in (0,T] \quad \text{and} \quad u_\ell(\omega, 0) = u_0(\omega)$$

with $\phi_\ell \in V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0}$ and

$$\langle \underline{A}_\ell(\omega) u_\ell(\omega), \phi_\ell \rangle_{\mathrm{L}^2(D)} = \langle \operatorname{div} \mathbf{F}(\omega, u_\ell(\omega)), \phi_\ell \rangle_{\mathrm{L}^2(D)} + \sum_{F \subset \mathcal{F} \setminus \Gamma_{\mathrm{in}}} \langle \mathbf{n}_{K,F} \cdot \Delta \mathbf{F}_{K,F}(\omega, u_\ell(\omega)), \phi_\ell \rangle_{\mathrm{L}^2(F)}$$

with $\Delta\mathbf{F}_{K,F}(\omega, u_\ell(\omega)) \coloneqq \mathbf{F}^*_{K,F}(\omega, \mathbf{u}_\ell(\omega)) - \mathbf{F}(\omega, \mathbf{u}_{\ell,K}((\omega)))$, the flux $\mathbf{F}(\omega, u(\omega)) = \mathbf{q}(\omega) u(\omega)$ and the numerical flux

$$\mathbf{F}^*_{K,F}(\omega, u_\ell(\omega)) = \begin{cases} \mathbf{F}(\omega, u_K(\omega)) & \mathbf{q}(\omega) \cdot \mathbf{n}_K > 0 \wedge F \in \mathcal{F} \setminus \partial D \\ \mathbf{F}(\omega, u_{K'}(\omega)) & \mathbf{q}(\omega) \cdot \mathbf{n}_K < 0 \wedge F \in \mathcal{F} \setminus \partial D \\ 0 & F \in \mathcal{F} \cap \partial D \setminus \Gamma_{\mathrm{in}} \end{cases}.$$

We refer to 2.32 for the deterministic and discrete version of the problem.

Assuming that the true solution provides sufficient regularity and that we can bound for a fixed $\omega \in \Omega$

$$\sum_{K \in \mathcal{K}} \sum_{F \in \mathcal{F}_K} \| \mathbf{n}_{K,F} \cdot \Delta \mathbf{F}_{K,F}(\omega, \boldsymbol{\phi}_\ell) \|^2_{\mathrm{L}^2(F)} \leq C_{\underline{A}}(\omega) \, \langle \underline{A}_\ell(\omega) \boldsymbol{\phi}_\ell, \boldsymbol{\phi}_\ell \rangle_{\mathrm{L}^2(D)}, \quad \forall \boldsymbol{\phi}_\ell \in V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0}$$

almost surely, we can apply theorem 2.31 to conclude semi-discrete convergence. However, examining the regularity of the solution with respect to random flux field remains an open challenge for theoretical investigations. Nevertheless, we will conduct experiments investigating the regularity of the problem with respect to the input data. To this end, we consider the following default configurations.

Problem Configuration 4.13. We consider the strong formulation where we search for the solution $u \colon \Omega \times [0,1]^2 \times [0,1] \to \mathbb{R}$ such that

$$\partial_t u(\omega, \mathbf{x}, t) + \operatorname{div} \mathbf{q}(\omega, \mathbf{x}) u(\omega, \mathbf{x}, t) = 0, \quad u(\omega, \mathbf{x}, 0) = u_0(\omega, \mathbf{x}).$$

The random initial data is modeled with the points $\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3 \in \mathbb{R}^2$ where the first component of these points is a uniformly distributed random variable $\mathcal{U}(0.25, 0.75)$ and the second component is 0.8. The function of the initial data is given by

$$u_0(\omega,\mathbf{x}) = \begin{cases} \max\limits_{\substack{\mathbf{c}\in\{\mathbf{c}_1,\mathbf{c}_2,\mathbf{c}_3\}\\ \|\mathbf{x}-\mathbf{c}\|<1/6}} \left(\cos\left(6\pi\,\|\mathbf{x}-\mathbf{c}(\omega)\|\,\right)+1\right)^2, & \bigvee_{i=1}^{3}\|\mathbf{x}-\mathbf{c}_i\|<1/6 \\ 0, & \text{else} \end{cases}.$$

In some experiments we also neglect the random initial data and only consider a mass concentration at $(0.5, 0.8)^\top$ with the same shape as above. Furthermore, the random flux field $\mathbf{q}$ is generated by solving the elliptic configuration 4.5 on log-normal fields, i.e., we considered $\sigma = 1.0$, $\boldsymbol{\lambda} = (0.15, 0.15)^\top$ and $\nu = 1.8$ and solved the system with mixed finite elements to ensure flux preservation.

Two samples of the above configuration are presented in figure 4.17. Each column depicts the numerical solution at times $t \in \{0.0,\ 0.25,\ 0.5,\ 0.75,\ 1.0\}$ from top to bottom. The first column shows the spatio-temporal development of the first sample on level $\ell = 8$ where the second column shows the same sample solved on level $\ell = 7$. Both computations are needed in equation (3.17).

Method Configuration 4.14. We search for numerical solutions in $V_{\ell,\mathbf{p}=2}^{\mathrm{dG}}$ using the numerical upwind flux and initialize the BMLMC method with $\{M_{0,\ell}\}_{\ell=0}^{L} = \left\{M_{0,4} = 2^{10},\ M_{0,5} = 2^{8},\ M_{0,6} = 2^{6},\ M_{0,7} = 2^{4}\right\}$. The computation is deployed on the HoReKa supercomputer for $\mathrm{T} = 6$ hours using $|\mathcal{P}| = 256$ processes. The hyperparameters of the method are chosen again as $\theta = 0.5$ and $\eta = 0.9$. The semi-discrete system is solved with an implicit midpoint rule where the time-step size is chosen for each sample with $\tau_\ell = 2^{-(\ell+2)}$. We refer to {41} for the complete configuration of all upcoming jobs. The default quantity of interest is the total mass at time T, i.e., the L^1-norm since the solution is positive.

Problem Experiments 4.15. We start again by tweaking the covariance function of the log-normal field inserted into the elliptic model problem. Since the flux used in default configuration 4.13 is given by $\mathbf{q}(\omega,\mathbf{x}) = -\kappa(\omega,\mathbf{x})\nabla u^{\mathrm{elliptic}}(\omega,\mathbf{x})$, we examine how the regularity of the random field $\kappa(\omega,\mathbf{x})$ is transmitted to the random flux field $\mathbf{q}(\omega,\mathbf{x})$ and thus to the final solution of the transport equation $u^{\mathrm{transport}}(\omega,\mathbf{x},t)$.

Starting with two experiments of a varying $\sigma \in \{0.5, 0.75, 1.0\}$ in figure 4.18 and a varying correlation length $\boldsymbol{\lambda} \in \left\{(0.05,0.05)^\top, (0.1,0.1)^\top, (0.15,0.15)^\top\right\}$ in figure 4.19, we observe that, as in the elliptic case, σ and $\boldsymbol{\lambda}$ do not influence the regularity of the solution and thus the convergence rate of the method. However, they do influence the hidden constant.

Next, we drop the randomness in the initial data and examine the smoothing parameter ν of the covariance function and how it influences the final solution of the transport problem. The results are given in figure 4.20. By the convergence plot in the lower right we can clearly see that the regularity of the random field $\kappa(\omega,\mathbf{x})$ indeed transmits all the way to the solution of the transport equation. A theoretical argument for this can be sketched by combining the left result of table 4.1 for the $|\cdot|_{\mathrm{H}^1(D)}$ semi-norm and considering the smooth initial conditions, i.e., the solution maintains its smoothness up to the end time since no further disturbances have been added by a rough flux field.

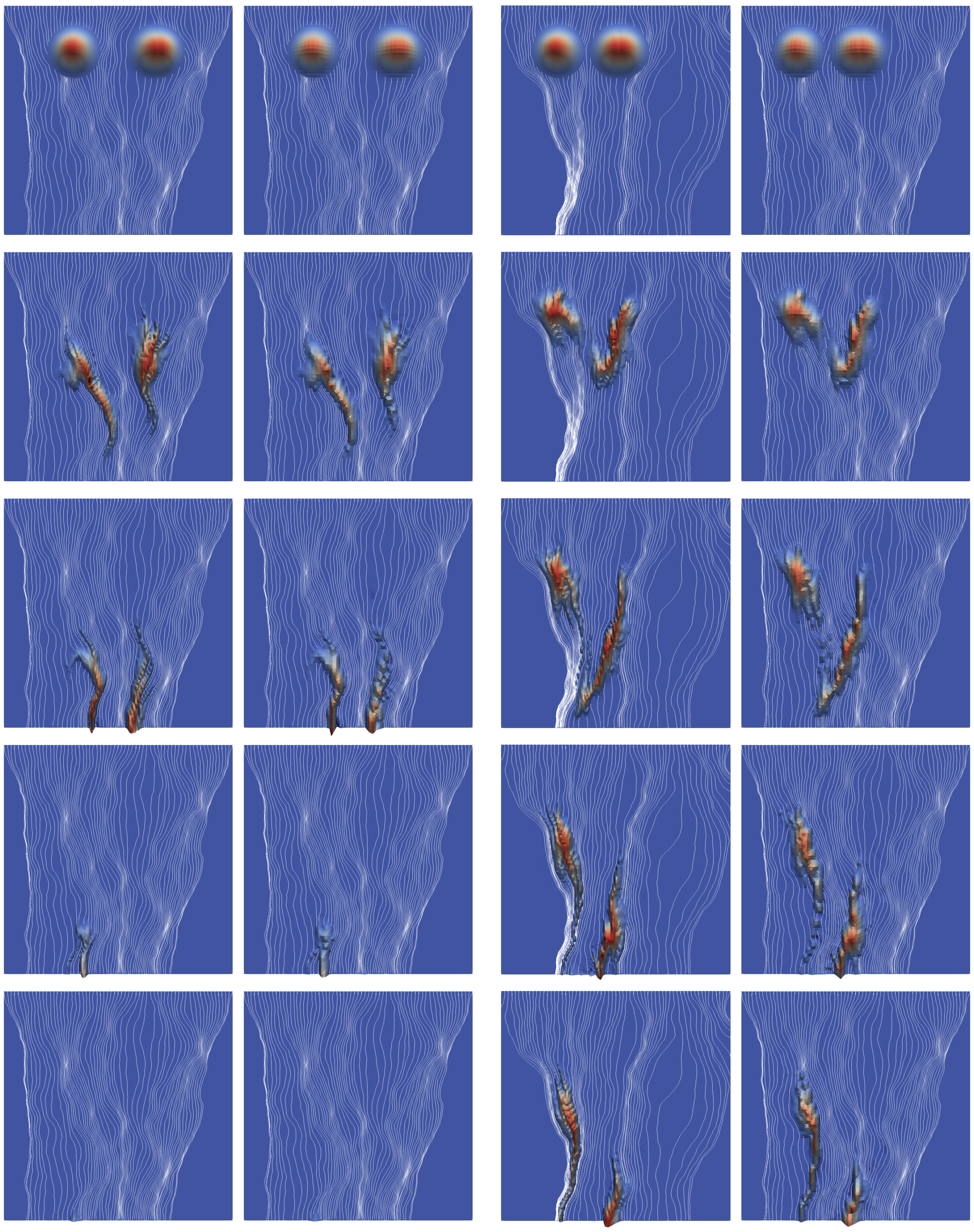


FIGURE 4.17: Two samples of configuration 4.13 each with a fine and a coarse mesh and time resolution. The shown time points are $t \in \{0.0,\ 0.25,\ 0.5,\ 0.75,\ 1.0\}$.

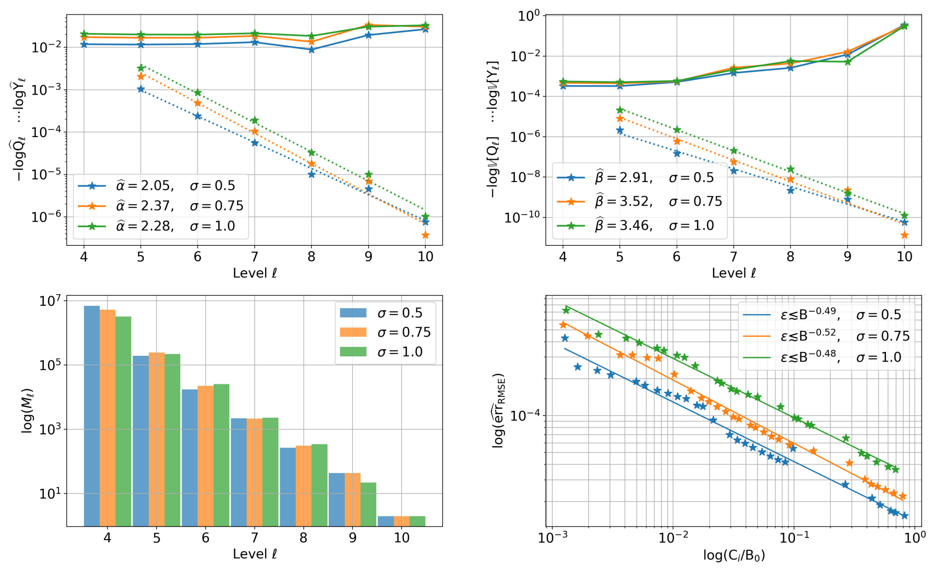


FIGURE 4.18: BMLMC method with configuration 4.14 applied to the transport problem configuration 4.13 for varying variance of the covariance function.

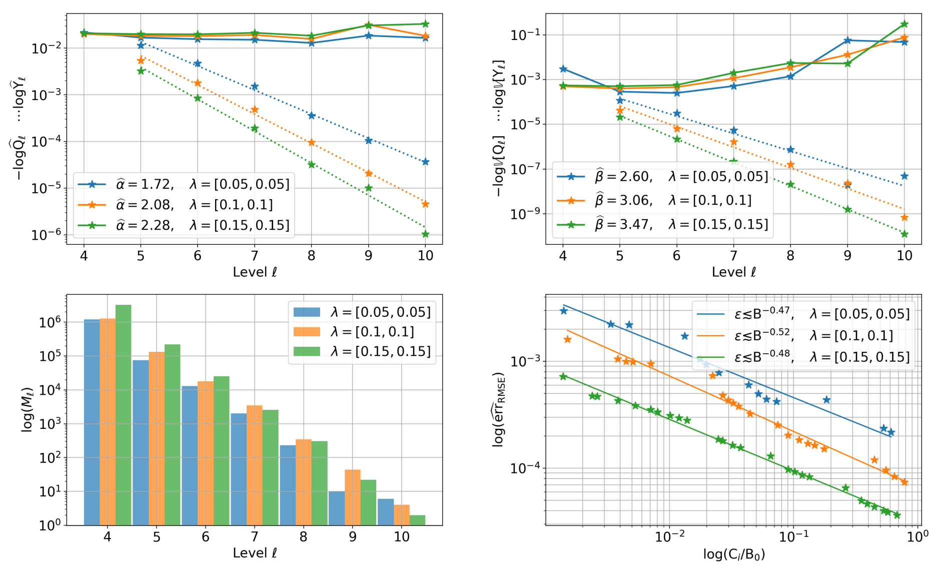


FIGURE 4.19: BMLMC method with configuration 4.14 applied to the transport problem configuration 4.13 with varying correlation length of the covariance function.

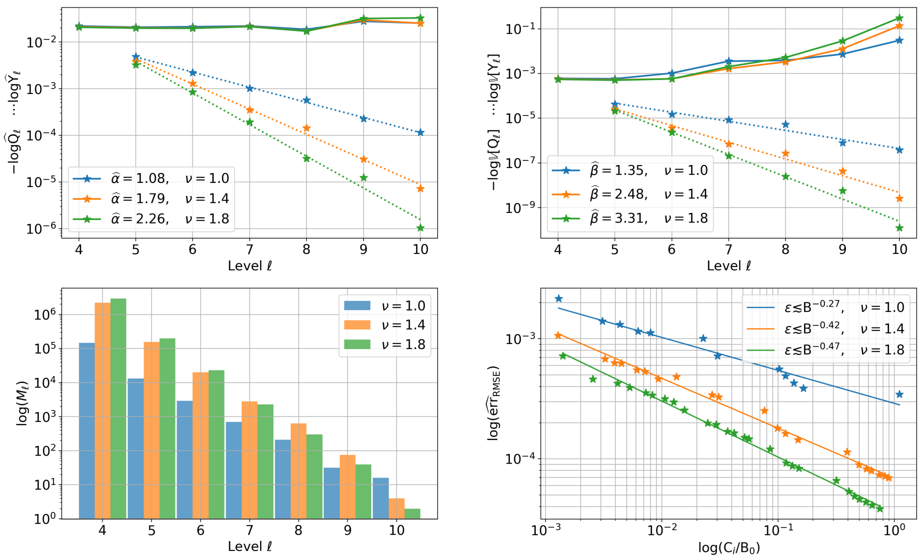


FIGURE 4.20: BMLMC method with configuration 4.14 applied to the transport problem configuration 4.13 for varying smoothness of the covariance function.

Method Experiments 4.16. Next, we investigate the relation between the time step size τ_ℓ and the mesh width h_ℓ. As described in [141], finding the right ratio $C_{\mathrm{CLF}} = \tau_\ell / h_\ell$ is crucial to design a fast overall method. If the ratio is too small, too many time steps are used. Likewise, if the ratio is too large, the arising system which is to solve each time step is worse conditioned, leading to a slower overall method. Therefore, we run the simulation with $\tau_\ell = C_{\mathrm{CFL}} \cdot 2^{-\ell}$ for different constants. The results are shown in figure 4.21. As we are in a budgeted framework, we can deduce by this figure that choosing $C_{\mathrm{CFL}} = 0.25$ leads to smallest $\mathrm{err}_{\mathrm{RMSE}}$ and thereby, we fix this ratio for all levels.

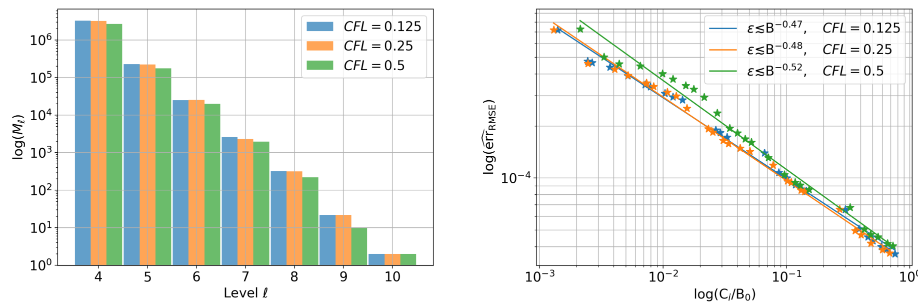


FIGURE 4.21: BMLMC method with configuration 4.14 applied to the transport problem configuration 4.13 for varying time step sizes.

Method Experiments 4.17. We shortly outline the experiment of figure 4.22 where we used different polynomial degrees for the ansatz functions of the dG method. We observe that for the given default problem, even though the smoothing parameter was chosen as $\nu = 1.8$, a higher degree is not worth to consider. As in the elliptic case, using a more expensive discretization even leads to a worse estimated RMSE due to the cost restrictions. We suspect that the reason for this is that the solution is very localized and that the higher polynomial degree adds additional degrees of freedom where they are not necessary. This experiments motivates to develop **p**-adaptive dG discretizations in future work to only use the higher polynomial degree where it is actually beneficial to the computation.

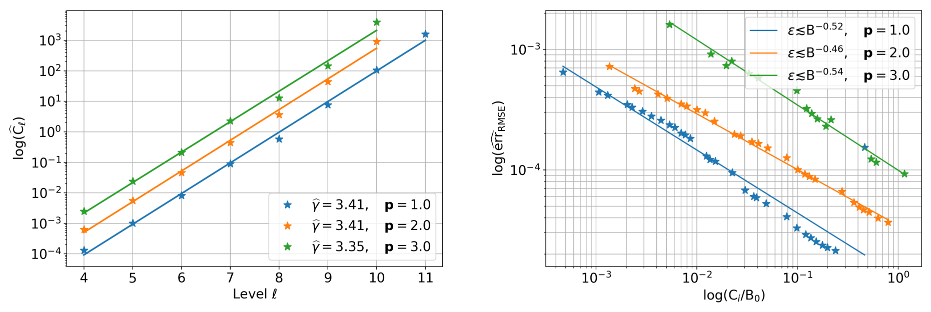


Figure 4.22: BMLMC method with configuration 4.14 applied to the transport problem configuration 4.13 for varying polynomial degree **p** of the dG ansatz funcitons.

Method Experiments 4.18. Last but not least, we investigate the proposed parallelism for this model problem. To this end, we compare the naive parallelization again with the multi-sample system parallelization. The interpretation of the previous section 4.1 is still valid, however, the experiment was conducted with $|\mathcal{P}| = 256$ processing units and two hours of total computing time. We note again, that the constant in the convergence result 3.54 is heavily dependent on the parallelism.

Next, we investigate the weak scaling of the parallelization. We conduct the experiment again with the idea of (3.34), i.e., we investigate the error reduction by adding further processing units. First, we notice by the plot on the left of figure 4.24 that: the more processing units are added, the more samples are computed on each level. Choosing comparable data points, we can deduce

$$\frac{\mathrm{err}_{\mathrm{RMSE}}^{|P|=512}}{\mathrm{err}_{\mathrm{RMSE}}^{|P|=128}} = 0.6932 \overset{!}{=} 4^{e} \quad \Leftrightarrow \quad e = -0.26.$$

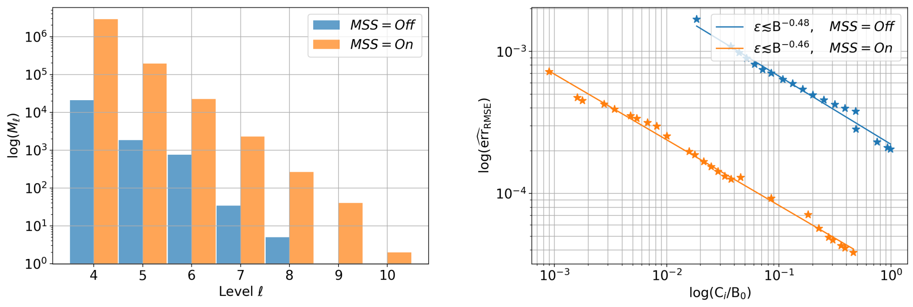


FIGURE 4.23: BMLMC method with configuration 4.14 applied to the transport problem configuration 4.13. Comparison of MSS parallelism with naive parallelization.

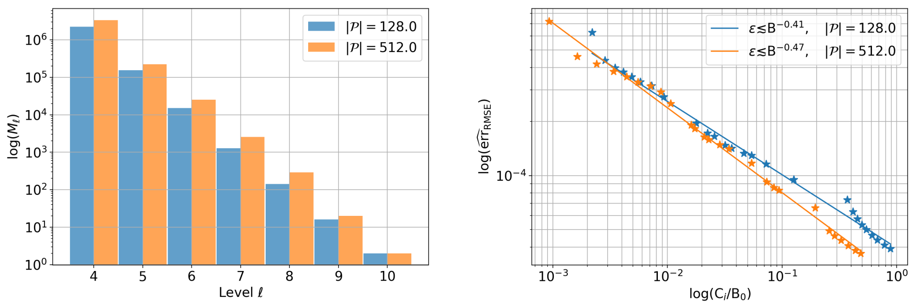


FIGURE 4.24: BMLMC method with configuration 4.14 applied to the transport problem configuration 4.13 while measuring the weak scaling of the parallelism.

Problem	Parameter Choice	Method	Parameter Choice	$(\widehat{\alpha}, \widehat{\beta}, \widehat{\gamma})$	Convergence	RMSE	Figure
Transport Configuration 4.13	$\sigma = 0.5$	Configuration 4.14	default	$(2.05, 2.91, 3.49)$	$\mathrm{B}^{-0.49}$	$1.54 \cdot 10^{-5}$	4.18
	$\sigma = 0.75$			$(2.37, 3.52, 3.46)$	$\mathrm{B}^{-0.52}$	$2.22 \cdot 10^{-5}$	
	$\sigma = 1.0$			$(2.28, 3.46, 3.45)$	$\mathrm{B}^{-0.48}$	$3.65 \cdot 10^{-5}$	
Transport Configuration 4.13	$\lambda = (0.05, 0.05)$	Configuration 4.14	default	$(1.72, 2.60, 3.42)$	$\mathrm{B}^{-0.47}$	$2.18 \cdot 10^{-4}$	4.19
	$\lambda = (0.10, 0.10)$			$(2.08, 3.06, 3.42)$	$\mathrm{B}^{-0.52}$	$7.40 \cdot 10^{-5}$	
	$\lambda = (0.15, 0.15)$			$(2.28, 3.47, 3.45)$	$\mathrm{B}^{-0.48}$	$3.64 \cdot 10^{-5}$	
Transport Configuration 4.13	$\nu = 1.0$	Configuration 4.14	default	$(1.08, 1.35, 3.36)$	$\mathrm{B}^{-0.27}$	$3.44 \cdot 10^{-4}$	4.20
	$\nu = 1.4$			$(1.79, 2.48, 3.43)$	$\mathrm{B}^{-0.42}$	$6.95 \cdot 10^{-5}$	
	$\nu = 1.8$			$(2.26, 3.31, 3.46)$	$\mathrm{B}^{-0.47}$	$3.85 \cdot 10^{-5}$	
Transport Configuration 4.13	default	Configuration 4.14	$C_{\mathrm{CFL}} = 0.125$	$(2.27, 3.54, 3.48)$	$\mathrm{B}^{-0.47}$	$3.64 \cdot 10^{-5}$	4.21
			$C_{\mathrm{CFL}} = 0.25$	$(2.28, 3.46, 3.45)$	$\mathrm{B}^{-0.48}$	$3.67 \cdot 10^{-5}$	
			$C_{\mathrm{CFL}} = 0.5$	$(2.25, 3.41, 3.42)$	$\mathrm{B}^{-0.52}$	$4.04 \cdot 10^{-5}$	
Transport Configuration 4.13	default	Configuration 4.14	$\mathbf{p} = 1$	$(1.72, 1.87, 3.41)$	$\mathrm{B}^{-0.52}$	$1.54 \cdot 10^{-4}$	4.22
			$\mathbf{p} = 2$	$(2.26, 3.33, 3.41)$	$\mathrm{B}^{-0.46}$	$3.67 \cdot 10^{-5}$	
			$\mathbf{p} = 3$	$(2.26, 3.12, 3.35)$	$\mathrm{B}^{-0.54}$	$9.25 \cdot 10^{-5}$	
Transport Configuration 4.13	default	Configuration 4.14	MSS Off, T$=$2h	$(2.67, 5.34, 1.07)$	$\mathrm{B}^{-0.48}$	$2.05 \cdot 10^{-4}$	4.23
			MSS On, T$=$2h	$(2.26, 3.31, 3.44)$	$\mathrm{B}^{-0.46}$	$3.86 \cdot 10^{-5}$	
Transport Configuration 4.13	default	Configuration 4.14	$\lvert\mathcal{P}\rvert = 128$, T$=$2h	$(2.22, 3.81, 3.37)$	$\mathrm{B}^{-0.41}$	$3.93 \cdot 10^{-5}$	4.24
			$\lvert\mathcal{P}\rvert = 512$, T$=$2h	$(2.27, 3.45, 3.48)$	$\mathrm{B}^{-0.47}$	$3.69 \cdot 10^{-5}$	

Table 4.3: Overview on all numerical results considering the transport model problem.

4.3 Experiments on the Acoustic Wave Equation

We recall the model problem 1.3. In this last section, we discuss the application of implicit time-stepping methods and space-time discretizations in combination with the BMLMC method and dG discretizations. This last problem is computationally extremely challenging due to many sources of uncertainty and the complex discretization procedure. We refer to section 2.3.4 for the discrete model problem and to section 3.3.4 for the extension on random fields. In particular, we proceed as described in 3.3.4 and employ log-normal fields for ρ and consider κ to be uniformly constant in the whole domain. Hence, the mass operator $M(\omega)$ and the flux operator $A(\omega)$ inherit the randomness and therefore, we consider the following semi-discrete model problem.

Model Problem 4.19. For a fixed $\omega \in \Omega$, we search for an approximate solution $\mathbf{u}_\ell(\omega) \in V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0}$ using dG elements where we assume that the discrete operator satisfies $\underline{M}_\ell(\omega) \in \mathcal{L}(V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0}, V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0})$ almost surely and that we have cell-wise constant wave speeds $c_K(\omega) = \sqrt{\kappa_K/\rho_K(\omega)}$. The right-hand side is deterministically chosen $\mathbf{b}_\ell \in \mathrm{L}^2((0,T), V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0})$ such as in 2.3.4. The matrix $\underline{M}_\ell(\omega)$ is determined by a Galerkin approximation, i.e.,

$$\langle \underline{M}_\ell(\omega)\mathbf{u}_\ell(\omega), \boldsymbol{\phi}_\ell\rangle_{\mathrm{L}^2(D)} = \langle M(\omega)\mathbf{u}_\ell(\omega), \boldsymbol{\phi}_\ell\rangle_{\mathrm{L}^2(D)}\,, \qquad \boldsymbol{\phi}_\ell \in V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0}.$$

The random flux operator is again constructed with upwind flux, i.e., for $\mathbf{u}_\ell(\omega) = (\mathbf{v}_\ell(\omega), p_\ell(\omega))^\top \in V^{\mathrm{dG}}_{K,\mathbf{p}\geq 0}$ and $\boldsymbol{\phi}_\ell = (\boldsymbol{\varphi}_\ell, \psi_\ell)^\top \in V^{\mathrm{dG}}_{K,\mathbf{p}\geq 0}$, we define $\underline{A}_\ell(\omega) = \sum_{K\in\mathcal{K}} \underline{A}_{\ell,K}(\omega) \in \mathcal{L}(V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0}, V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0})$ by

$$\begin{aligned}\left\langle \underline{A}_{\ell,K}(\omega)\mathbf{u}_\ell(\omega), \boldsymbol{\phi}_\ell\right\rangle_{\mathrm{L}^2(K)} &= -\left\langle \nabla p_{\ell,K}(\omega), \boldsymbol{\varphi}_{\ell,K}\right\rangle_{\mathrm{L}^2(K)} - \langle \operatorname{div} \mathbf{v}_{\ell,K}(\omega), \psi_{\ell,K}\rangle_{\mathrm{L}^2(K)}\\ &- \sum_{F\in\mathcal{F}_K} \frac{1}{Z_K(\omega) + Z_{K'}(\omega)} \left\langle [\![p_{\ell,K}(\omega)]\!]_F + Z_{K'}(\omega)[\![\mathbf{v}_{\ell,K}(\omega)]\!]_F \cdot \mathbf{n}_{K,F},\, \psi_{\ell,K} + Z_K(\omega)\boldsymbol{\varphi}_{\ell,K}\cdot \mathbf{n}_{K,F}\right\rangle_{\mathrm{L}^2(F)}\end{aligned}$$

where $Z_K(\omega) = \sqrt{\kappa_K \rho_K(\omega)}$ is a cell-wise random impedance.

Assuming again that the true solution provides sufficient regularity and that we can bound for a fixed $\omega \in \Omega$

$$\sum_{K\in\mathcal{K}} \sum_{F\in\mathcal{F}_K} \|\mathbf{n}_{K,F}\cdot \Delta\mathbf{F}_{K,F}(\omega,\boldsymbol{\phi}_\ell)\|^2_{\mathrm{L}^2(F)} \leq C_{\underline{A}}(\omega)\,\langle \underline{A}_\ell(\omega)\boldsymbol{\phi}_\ell, \boldsymbol{\phi}_\ell\rangle_{\mathrm{L}^2(D)}\,, \quad \forall \boldsymbol{\phi}_\ell \in V^{\mathrm{dG}}_{\ell,\mathbf{p}\geq 0}$$

almost surely, we can also apply theorem 2.31 to conclude semi-discrete convergence for this problem. Using theorem 2.38 and under sufficient regularity assumptions, we can deduce convergence for the space-time discretization, too. However, we emphasize again that for predicting the regularity of the true solution with respect to the random material parameters more analytical investigations have to be done.

Problem Configuration 4.20. We consider the time interval $[0, 1]$, deterministic and homogeneous initial conditions $(\mathbf{v}_0, p_0) = (\mathbf{0}, 0)^\top$ and a constant compression module $\kappa(\omega, \mathbf{x}) \equiv 1$ on the domain $D = [0, 1]^2$. The right-hand sides are given by $\mathbf{f} \equiv \mathbf{0}$ and $g(\mathbf{x}, t) = 10\, g_1(t)\, g_2(\mathbf{x})$. The function $g_1(t)$ is a Ricker-wavelet, i.e., with $a = \frac{\pi}{10}$, it is given by

$$g_1(t) = \left(1 - \left(\tfrac{t}{a}\right)^2\right) \cdot \exp\left(-\tfrac{t^2}{2a^2}\right), \quad t \in [0,1].$$

The function $g_2(\mathbf{x})$ is a nascent delta function of width $w = 0.1$, i.e., with an appropriate constant η_{g_2} such that $\|g_2\|_{\mathrm{L}^1(D)} = 1$ and the center $\mathbf{c} = (0.5, 0.75)^\top$, it is given by

$$g_2(\mathbf{x}) = \begin{cases} \eta_{g_2} \exp\left(-\frac{1}{1-\left\|\frac{\mathbf{x}-\mathbf{c}}{w}\right\|^2}\right) & \|\mathbf{x}-\mathbf{c}\| < w \\ 0 & \|\mathbf{x}-\mathbf{c}\| \geq w \end{cases}.$$

Furthermore, we have employed log-normal fields as in the previous experiments with $\sigma = 1.0$, $\boldsymbol{\lambda} = (0.15, 0.15)^\top$ and $\nu = 1.8$. The default quantity of interest is the L^2-norm in the region of interest $D^{\mathrm{RoI}} = [0.25, 0.75] \times [0, 0.25]$ at time $T = 1$.

Method Configuration 4.21. We search for numerical solutions in $V_{\ell,\mathbf{p}=2}^{\mathrm{dG}}$ and start the BMLMC method with $\theta = 0.5$, $\eta = 0.9$ and $\{M_{0,\ell}\}_{\ell=0}^{L} = \{M_{0,4} = 2^{10}, M_{0,5} = 2^8, M_{0,6} = 2^6, M_{0,7} = 2^4\}$. The experiments are conducted on the HoReKa supercomputer for T = 6 hours using $|\mathcal{P}| = 1024$ processing units. The semi-discrete system is solved using an implicit midpoint rule with the time step size $\tau = 2^{-\ell+3}$. The configuration of the benchmark jobs is given in {42}.

Method Configuration 4.22. We also consider a space-time discretization using the dG ansatz space $V_{\ell,\mathbf{p}_K=1,\mathbf{p}_I=1}^{\mathrm{ST-dG}}$ in combination with the BMLMC method on $|\mathcal{P}| = 512$ processing units for two space dimensions and on $|\mathcal{P}| = 64$ for one space dimension. Furthermore, the computation times are T = 10 minutes and T = 12 hours for one and two dimensions, respectively. The space-time system is solved with the GMRES method and a point-block Gauss-Seidel preconditioner. The technical configuration is given in {43}.

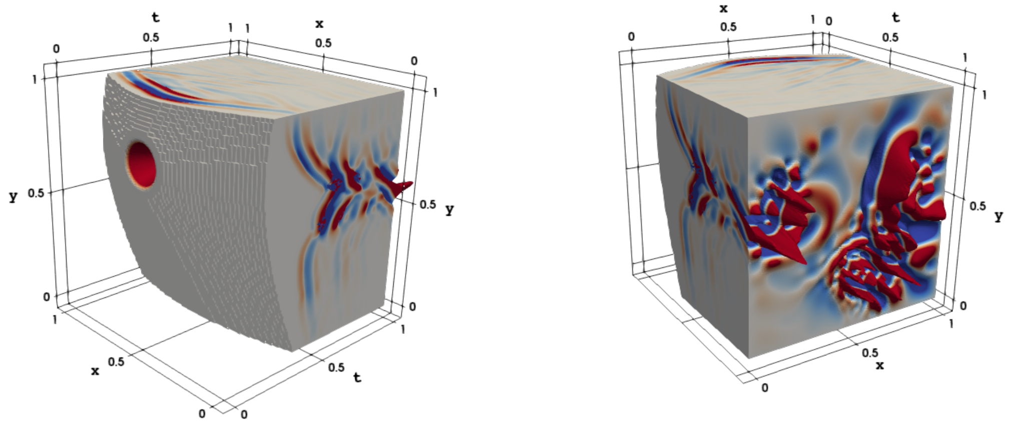
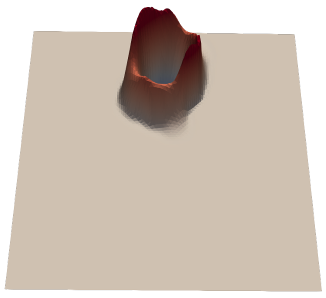
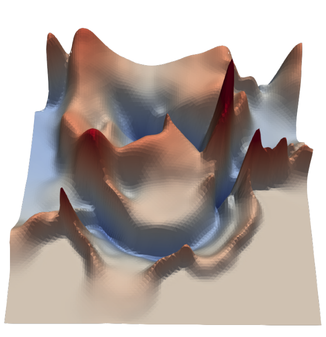
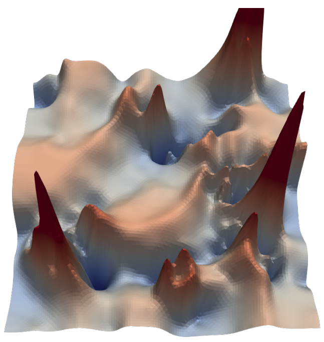


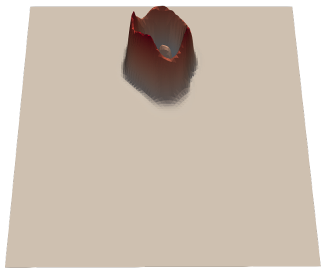
Figure 4.25: Space-time cube seen from the start $t = 0.0$ (left) and the end $t = 1.0$ (right).

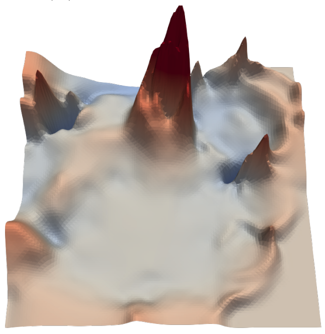
The problem configuration 4.20 is illustrated in figure 4.25 using the space-time discretization of configuration 4.22. The figure shows a space-time cube seen from the first and the last time point. Additionally, the problem configuration 4.20 is given in figure 4.26 using the implicit midpoint rule and configuration 4.21. This figure illustrates the approximations at the time points $t = 0.0625$, $t = 0.5$ and $t = 1.0$ of two different realizations $m = 1$ and $m = 2$.


(a) $m = 1$ and $t = 0.0625$


(b) $m = 1$ and $t = 0.5$


(c) $m = 1$ and $t = 1.0$


(d) $m = 2$ and $t = 0.0625$


(e) $m = 2$ and $t = 0.5$

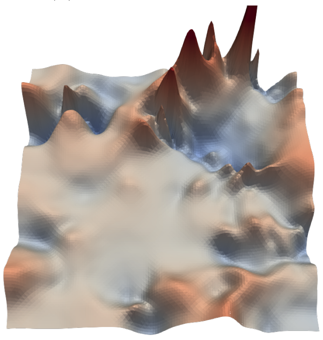

(f) $m = 2$ and $t = 1.0$

FIGURE 4.26: Comparison of the temporal development of two sample solutions $m = 1$ and $m = 2$ to the acoustic wave equation propagating through random media.

Problem Experiments 4.23. As we have seen in section 4.1 and 4.2 the structure of the log-normal field has a big influence on the constant and the convergence rate of theorem 3.54. We will conduct a last investigation of the log-normal field by adjusting once again σ in the covariance kernel. The results of this experiment are given in figure 4.27. We see that we are in the case $\gamma > \beta$ for this problem. Hence, the computational cost is dominated by the highest level (confer the center right plot of figure 4.27). We further remark that the BMLMC method works very reliably for this model problem and is capable to exhaust the large computational budget of $\mathrm{B} = 1024 \cdot 6$ CPU hours completely.

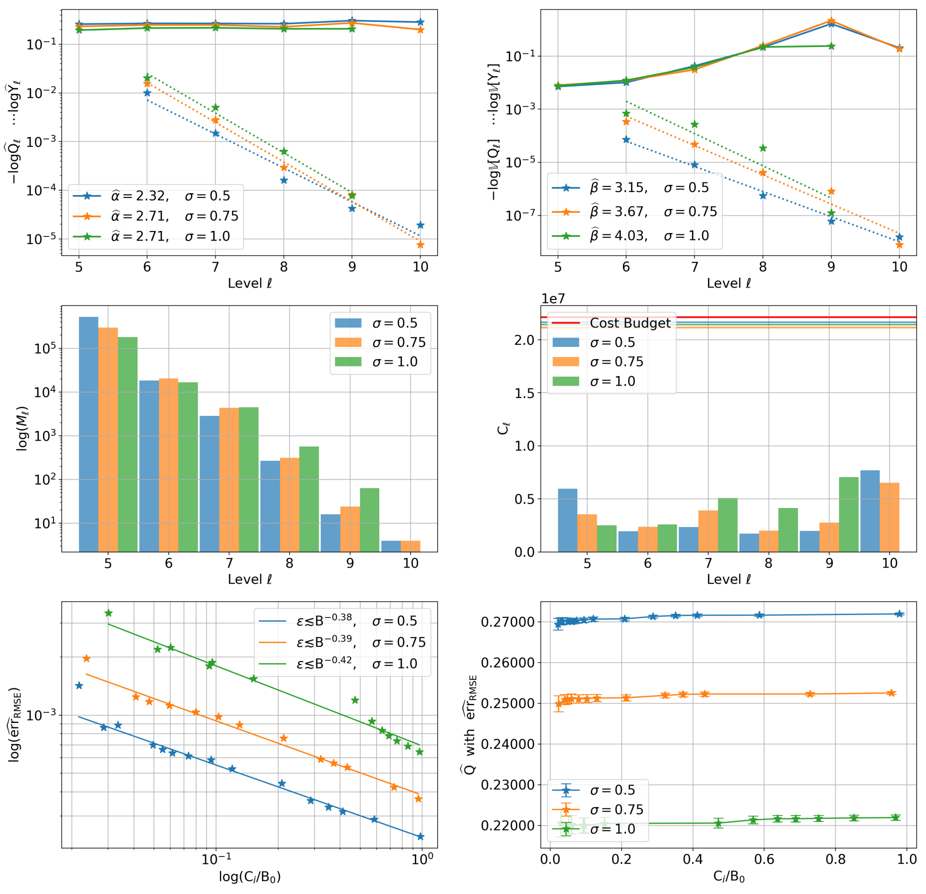


Figure 4.27: BMLMC method with problem and method configuration 4.20 applied to the acoustic problem configuration for varying variance of the covariance function.

Method Experiments 4.24. First, we search again for the optimal ratio τ_ℓ/h_ℓ. The experiment is shown in figure 4.28 from which we can conduct that $C_{\mathrm{CFL}} = 0.125$ yields the best results.

Next, we investigate the polynomial degree in figure 4.29. While we did not find any advantages in using higher polynomial degrees for the previous model problems, we clearly see it for configuration 4.20. Even though the method is not able to utilize the complete computational budget for all degrees, we can deduce that degree $\mathbf{p} = 2$ or $\mathbf{p} = 3$ should be chosen in order solve this model problem.

We also investigate different implicit time-stepping schemes with a global convergence order of $\mathcal{O}(\tau^2)$. In particular, we consider the Butcher tableaus, i.e., the configuration of a_{ij}, b_i and c_i (confer (2.22))

$$\begin{array}{c|c} 1/2 & 1/2 \\ \hline & 1 \end{array} \qquad \begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1/2 & 1/2 \\ \hline & 1/2 & 1/2 \end{array} \qquad \begin{array}{c|cc} 1/4 & 1/4 & 0 \\ 3/4 & 1/2 & 1/4 \\ \hline & 1/2 & 1/2 \end{array}$$

which belong to the implicit midpoint rule (left), the Crank-Nicolson method (middle) and another diagonal implicit Runge-Kutte method (right). By the lower right plot of figure 4.30, we firstly notice that all methods give the same quantity of interest as expected. However, we also notice that by using the implicit midpoint rule, the BMLMC method is able to draw more samples and even an additional level resulting in the smallest error. We suspect that the reason for this is that the implicit midpoint rule only has one function evaluation in (2.22) instead of two. Thus, the BMLMC method can utilize this cost savings to draw more samples. Lastly, we mention that we experimented with explicit Runge-Kutta methods, too. The downside of explicit Runge-Kutta methods is that we have to decrease the time step size τ drastically in order to stabilize the computation uniformly over all samples. Locally adaptive schemes as in [122] might overcome this issue but further investigations have to be conducted.

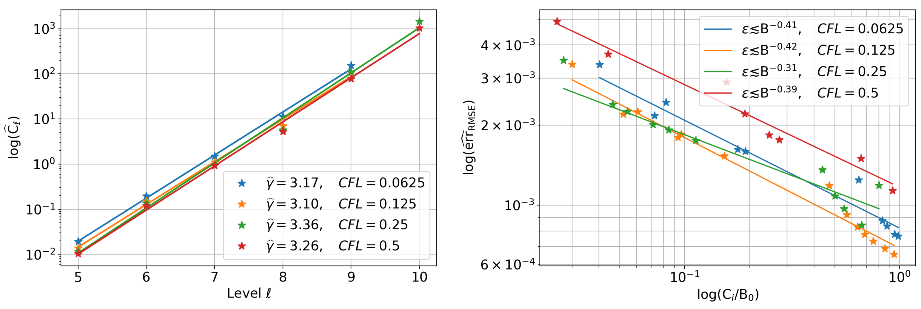


FIGURE 4.28: BMLMC method with configuration 4.21 applied to the acoustic problem configuration 4.20 for varying polynomial degree **p** of the dG ansatz funcitons.

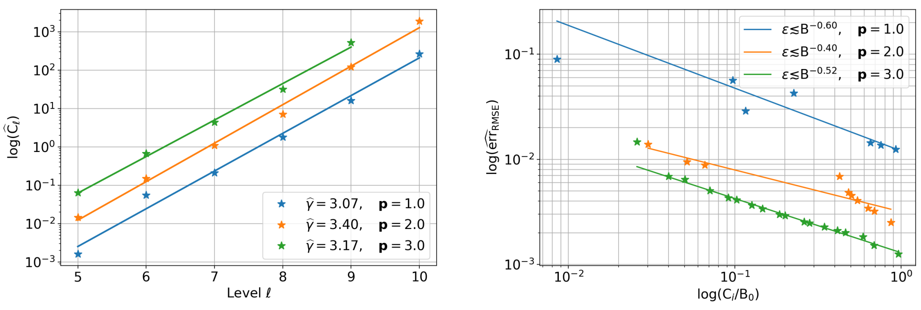


FIGURE 4.29: BMLMC method with configuration 4.21 applied to the acoustic problem configuration 4.20 for varying polynomial degree **p** of the dG ansatz funcitons.

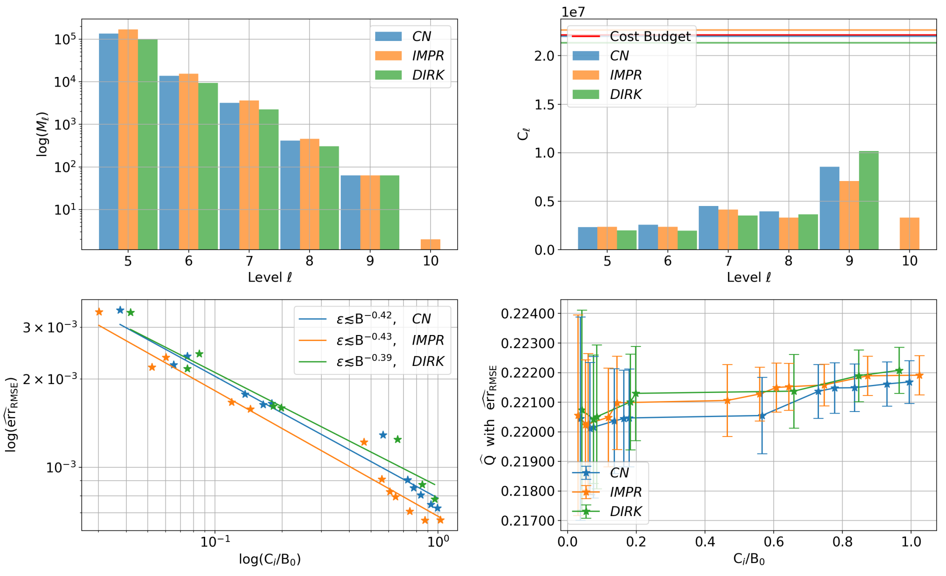


Figure 4.30: BMLMC method with configuration 4.21 applied to the acoustic problem configuration 4.20. Comparision of the implicit midpoint rule (IMPR), the Crank-Nicolson method (CR) and a diagonal implicit Runge-Kutta method (DIRK).

Method Experiments 4.25. Once again, we compare the naive parallelization with the multi-sample parallelism. The results are given in figure (4.31) from which we can draw the same conclusion as in the previous sections. Furthermore, we conducted a weak scaling experiment, i.e., increasing the computational ressources from $|\mathcal{P}| = 128$ to $|\mathcal{P}| = 2048$ while keeping the computational time fixed to $\mathrm{T} = 6\mathrm{h}$. Following (3.34), we can deduce with comparable data points

$$\frac{\mathrm{err}_{\mathrm{RMSE}}^{|P|=2048}}{\mathrm{err}_{\mathrm{RMSE}}^{|P|=128}} = 0.3893 \overset{!}{=} 16^{e} \quad \Leftrightarrow \quad e = -0.34.$$

By figure (4.31) on the right, we can observe that the BMLMC method in combination with the multi-sample systems utilizes the additional computing ressources effectively for error reduction and thereby scales weakly very well.

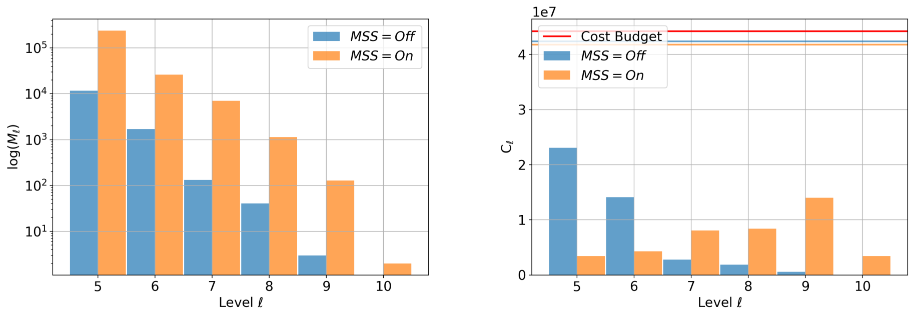


FIGURE 4.31: BMLMC method with configuration 4.21 applied to the acoustic problem configuration 4.20. Comparison of MSS parallelism with naive parallelization.

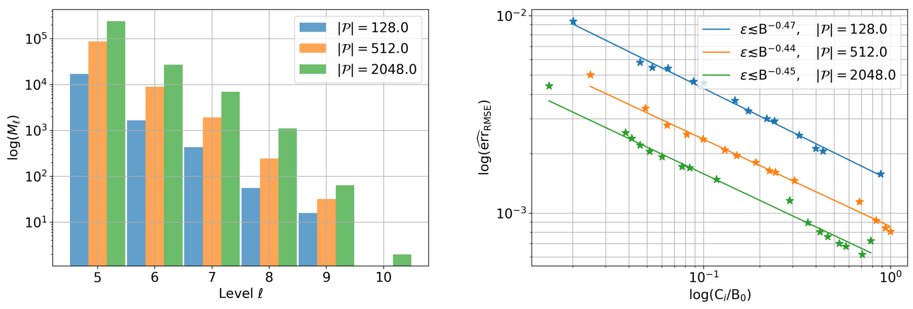


FIGURE 4.32: BMLMC method with configuration 4.21 applied to the transport problem configuration 4.20 while measuring the weak scaling of the parallelism.

Problem Configuration 4.26. We consider a one-dimensional space-time problem on an open spatial domain $D = (0, 1)$ and a random material density $\rho\colon \Omega \times \overline{D} \to \mathbb{R}$ which is in $[0.25, 0.75]$ modeled with one-dimensional log-normal fields generated with the ciruclant embedding method with the default parameters of 4.20. For the rest of the domain we choose $\rho \equiv 0.5$. The quantity of interest is the L^2-norm over the whole space-time square. Furthermore, we consider homogeneous right-hand sides and the initial conditions

$$p_0(x) = \begin{cases} \cos^2(5\pi x) & |x| < 0.1 \\ 0 & |x| \geq 0.1 \end{cases}, \qquad v_0(x) = -p_0(x).$$

Method Experiments 4.27. We conduct experiments with the space-time discretization in one and two space dimensions. For the one-dimensional problem we consider the configuration 4.26. The results are presented in figure 4.33. We first note that $\gamma > \beta$ and that the cost is dominated by the highest level. We further observe again that σ influences the hidden constant and that the method delivers feasible results for

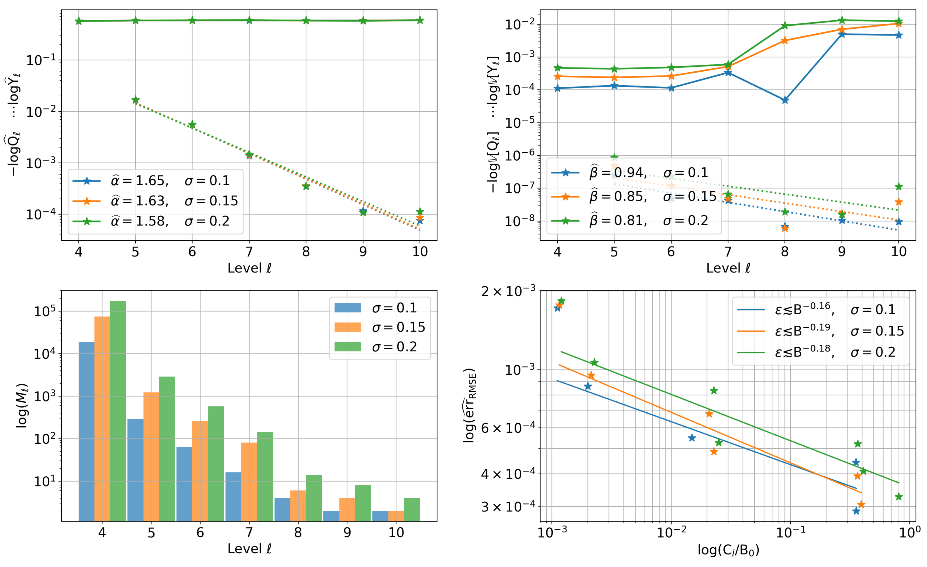


Figure 4.33: BMLMC method with configuration 4.22 applied to the acoustic problem configuration 4.26.

this configuration.

Moving on to the two-dimensional case given in figure 4.34 solving the problem configuration 4.20, we see by the lower left plot that the GMRES method with point-block Gauss-Seidel preconditioner still satisfies (3.15) for the multi-sample space-time system. However, the computational cost is rather high such that level nine can not be computed with the given configuration. Speeding up the linear solver with sufficient multi-grid preconditioning as proposed in [77, 62] enables to draw more samples and levels to further improve the estimates of the exponents. We leave the integration of a more performant preconditioner open for further investigations.

Outlook 4.28. In [190] the usage of a SC method for wave equations is proposed. We suspect that for a moderate parametric dimension and sufficient regularity a SC method outperforms the MLMC method. However, the application of the SC or even the MLSC is more difficult, and it is not obvious for which problem configuration the stochastic collocation yields better results. We have taken first steps towards incorporating stochastic collocation methods in our framework by integrating the library TASMANIAN [234] and distributing the collocation points on the parallel data structure. However, since we have not conducted an error analysis nor further numerical experiments we only present a proof of concept by figure 4.35. In this computation, we employed sparse grids to mimic random initial data, a dG discretization in space and the implicit midpoint rule. The figure illustrates the initial condition and two samples evaluated at the time points $t = 0.0625$, $t = 0.5$ and $t = 1.0$.

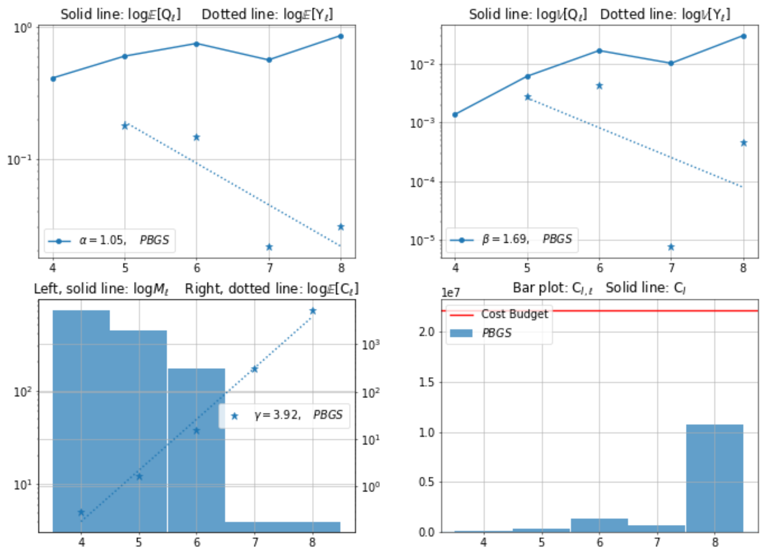
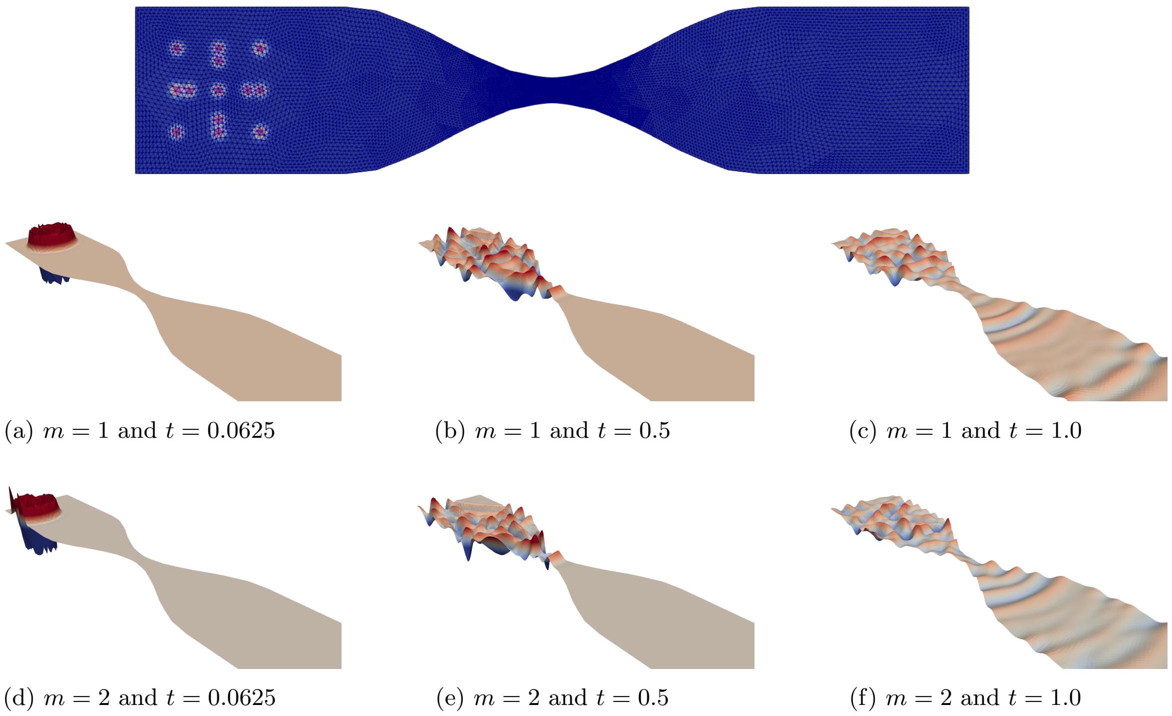


FIGURE 4.34: BMLMC method with configuration 4.22 applied to the acoustic problem configuration 4.20.



(a) $m = 1$ and $t = 0.0625$ (b) $m = 1$ and $t = 0.5$ (c) $m = 1$ and $t = 1.0$

(d) $m = 2$ and $t = 0.0625$ (e) $m = 2$ and $t = 0.5$ (f) $m = 2$ and $t = 1.0$

FIGURE 4.35: Acoustic wave equation with intial data modeled by sparse grids. Comparison of two samples $m = 1$ and $m = 2$ assoicated with different collocation points and illustration of the initial condition.

Problem	**Parameter Choice**	**Method**	**Parameter Choice**	$(\widehat{\alpha}, \widehat{\beta}, \widehat{\gamma})$	**Convergence**	**RMSE**	**Figure**
Acoustic Configuration 4.20	$\sigma = 0.5$	Configuration 4.21	default	$(2.32, 3.15, 3.55)$	$\mathrm{B}^{-0.38}$	$2.35 \cdot 10^{-4}$	4.27
	$\sigma = 0.75$			$(2.71, 3.67, 3.46)$	$\mathrm{B}^{-0.39}$	$3.68 \cdot 10^{-4}$	
	$\sigma = 1.0$			$(2.71, 4.03, 3.11)$	$\mathrm{B}^{-0.42}$	$6.45 \cdot 10^{-4}$	
Acoustic Configuration 4.20	default	Configuration 4.21	$C_{\mathrm{CFL}} = 0.0625$	$(2.71, 4.15, 3.17)$	$\mathrm{B}^{-0.41}$	$7.65 \cdot 10^{-4}$	4.28
			$C_{\mathrm{CFL}} = 0.125$	$(2.71, 4.04, 3.10)$	$\mathrm{B}^{-0.42}$	$6.55 \cdot 10^{-4}$	
			$C_{\mathrm{CFL}} = 0.25$	$(2.28, 2.67, 3.36)$	$\mathrm{B}^{-0.31}$	$1.19 \cdot 10^{-3}$	
			$C_{\mathrm{CFL}} = 0.5$	$(1.89, 2.34, 3.26)$	$\mathrm{B}^{-0.39}$	$1.14 \cdot 10^{-3}$	
Acoustic Configuration 4.20	default	Configuration 4.21	$\mathbf{p} = 1$	$(1.51, 0.51, 3.07)$	$\mathrm{B}^{-0.60}$	$1.25 \cdot 10^{-2}$	4.29
			$\mathbf{p} = 2$	$(2.91, 3.75, 3.40)$	$\mathrm{B}^{-0.40}$	$2.50 \cdot 10^{-3}$	
			$\mathbf{p} = 3$	$(2.81, 5.44, 3.17)$	$\mathrm{B}^{-0.52}$	$1.26 \cdot 10^{-3}$	
Acoustic Configuration 4.20	default	Configuration 4.21	Crank-Nicolson	$(2.70, 4.01, 3.12)$	$\mathrm{B}^{-0.42}$	$7.27 \cdot 10^{-4}$	4.30
			Implicit MPR	$(2.60, 3.33, 3.34)$	$\mathrm{B}^{-0.43}$	$6.61 \cdot 10^{-4}$	
			DIRK	$(2.72, 4.15, 3.17)$	$\mathrm{B}^{-0.39}$	$7.78 \cdot 10^{-4}$	
Acoustic Configuration 4.20	default	Configuration 4.21	MSS Off	$(2.29, 5.07, 1.48)$	$\mathrm{B}^{-0.51}$	$1.26 \cdot 10^{-3}$	4.31
			MSS On	$(2.60, 3.22, 3.33)$	$\mathrm{B}^{-0.42}$	$7.26 \cdot 10^{-4}$	
Acoustic Configuration 4.20	default	Configuration 4.21	$\lvert\mathcal{P}\rvert = 128$, T = 6 hours	$(2.51, 4.37, 3.06)$	$\mathrm{B}^{-0.47}$	$1.58 \cdot 10^{-3}$	4.32
			$\lvert\mathcal{P}\rvert = 256$, T = 6 hours	$(2.62, 4.00, 3.16)$	$\mathrm{B}^{-0.44}$	$8.09 \cdot 10^{-4}$	
			$\lvert\mathcal{P}\rvert = 1024$, T = 6 hours	$(2.60, 3.15, 3.36)$	$\mathrm{B}^{-0.45}$	$7.26 \cdot 10^{-4}$	
1D space-time Problem 4.22	$\sigma = 0.1$	Configuration 4.26	default	$(1.65, 0.94, 3.59)$	$\mathrm{B}^{-0.16}$	$2.89 \cdot 10^{-4}$	4.33
	$\sigma = 0.15$			$(1.63, 0.85, 3.66)$	$\mathrm{B}^{-0.19}$	$3.05 \cdot 10^{-4}$	
	$\sigma = 0.2$			$(1.58, 0.81, 3.70)$	$\mathrm{B}^{-0.18}$	$3.27 \cdot 10^{-4}$	

Table 4.4: Overview on all numerical results considering the acoustic model problem.

5 Conclusion and Outlook

Man is a decision-making animal and, above all, conscious of this fact. This self-scrutiny has led to continuing efforts to find efficient ways of behaving in the face of complexity and uncertainty.
- Richard Bellman, 1966, [32]

Retrospective View

MLMC methods have been very successfully applied to many problems and a rich theory around the multi-level estimation idea was developed within the last decade. The main reason for this is arguably the simplicity of the assumptions 3.45 providing an easy and accessible interface to the application of the method. This PhD thesis started with the goal to develop multi-level Monte Carlo (MLMC) methods combined with suitable spatio-temporal discretizations for wave equations in random media. However, this turned out to be a challenging task because modeling highly heterogeneous random media requires a large amount of random variables and the solutions possess little regularity with respect to the material parameters. Out of these two reasons, the computational demand to approximate a solution is extremely high.

When we started with our investigations, there was already preliminary work for this particular application case in [185, 238], but we wanted to go beyond and combine the MLMC method with further spatio-temporal discretizations to achieve uniform stability over all samples. In particular, we wanted to be able to select from a collection of time-stepping methods and to employ discontinuous Galerkin and space-time discretizations as published in [141, 62, 75, 39]. To achieve this, we started to apply the MLMC method to simpler problems first such as the elliptic model of subsurface flows and the hyperbolic transport model in a randomly distributed flux field. The results of this preliminary work are published in [28]. However, we soon realized that moving from the simpler to the more challenging models requires a comprehensive framework to investigate arbitrary models with high-dimensional input data. In this thesis, we took first steps towards a unified application of methods from finite element theory and uncertainty quantification.

The extremely high computational demand resulting from the discretization in space and time as well as the high-dimensional parameter space motivated us to search for new ways to distribute the work load on a supercomputer. At this point, software development and high-performance computing became the main focus in this thesis. Developing the new parallelization following the simple idea of (2.4) required many changes in the preexisting code base. All parallelization aspects touched in chapter 2 essentially describe where we had to implement the changes in order to solve multiple samples in a singular distributed data-structure. Even though there have been approaches [16, 237, 79] to tackle the parallelization issue of MLMC methods, no solution has emerged to popularity yet. While we experimented with similar techniques at the beginning, an integrated solution soon proved to be reliable and more open to new features like integrating the SC method, the BMLMC method or automated benchmarking.

To move towards an integrated solution safely, effectively and efficiently, we also adapted the development workflow itself, i.e., we have explored continuous delivery and agile working methodologies for research software inspired by [6, 15, 88]. At the center of this workflow was the development of a continuous delivery pipeline which enabled us to deploy all computations completely automated on large computing clusters like the HoReKa supercomputer.

This workflow and the idea to empirically find the best algorithm combination to minimize (1.1) motivated the knapsack problem. This new point of view enabled the design of the final BMLMC method with dynamic programming techniques and ideas of the continuation MLMC [61] method. Our new method is theoretically motivated, numerically well-tested and has proven to perform extremely robust and reliable. We see it as tool "to find efficient ways of behaving in the face of complexity and uncertainty".

Conclusion

The main contributions and features of this thesis are:

- We demonstrate the applicability of MLMC methods to a wide selection of PDEs by utilizing various finite element and time discretization methods. Among these PDE models, we investigate the numerically highly challenging problem of approximating acoustic wave equations in random, heterogeneous media.

- We motivate, implement and test a new parallelism for the joint effort of UQ and FE methods. The parallelism is based on a single, massively distributed data-structure representing multiple samples discretized with finite element methods.

- We introduce a new workflow for developing and verifying high performance research software and conduct numerical experiments completely automated and repeatable via a continuous delivery pipeline.

- We propose a new budgeted variant of the MLMC which proves to be widely applicable. The BMLMC methods requires little a priori knowledge, shows to be reliable and robust, and delivers results in a performant and fully parallelized manner.

Outlook

Applications. To pick up on the initial goal of this thesis, i.e., to establish MLMC methods for wave equations, we still see a lot of potential for further research. In particular, approximating visco-elastic waves or Maxwell systems with the current framework is computationally achievable. The techniques proposed in [141, 75, 39, 62, 76] consider these new types of models already. However, solving these systems remains challenging due to their high computational and analytical complexity in three space-dimensions. We currently believe it is as a matter of scaling everything up while carefully extending the theory to these new models. Additionally, the framework can be extended to many other PDEs and answer questions for example arising in material sciences.

Applications of M++ which might benefit from additional uncertainty quantification are: enhancing cardio-vascular simulations [94, 102, 95] with probabilistic cell-models, combining the proposed UQ methods with interval arithmetics [261], developing uncertainty models for carbon capturing [150] or developing a Bayesian perspective for the preexisting full-wave inversion [39, 83]. All of these applications impose new theoretical questions but also further challenges for the software development.

Software. Developing software is an endless effort, it is never finished and improvements can always be implemented. We shortly present ideas on further features for M++: Effective preconditioning, especially for randomly distributed and high-dimensional systems, is essential for efficient computations. This is particularly true for space-time systems. Integrating a highly performant multi-grid preconditioner based on [62, 77] into the UQ application is an open issue but promises to further speed up the computations. Another improvement of the linear solver performance could be achieved by the integration of Ginkgo [7]. Having a clean interface to this solver library would allow to solve multi-sample systems on GPU clusters.

We further note that designing accessible interfaces for scientists applying these methods to their problems is key to the success of the software tool. We think that developing Python bindings wrapping the C++ code (like [197, 161]) can drastically improve the usability. Alternatively, computational methods including the hardware ressources could be provided by a software as a service (SaaS) model. This would expose the developed methods to a broader scientific community.

The development of budgeted MLSC, MLQMC or MIMC methods on the same distributed data-structure would enable to choose the best non-intrusive UQ method for a particular model by comparing the development of the estimated errors. This can include multi-level estimators for higher central moments as proposed in [155] or even conditional expectations and probability distributions [11] to be able to further characterize the output of highly complex and random systems.

UQ Methods. Gathering certainty comes with a cost. Thereby, we see great potential in linking UQ problems with problems of optimal ressource allocation. We think that the connection between UQ and dynamic programming/reinforcement learning can yield many new algorithms of high practical relevance. In particular, we suspect that the integrated parallelism and the BMLMC method can be utilized to solve optimal control problems with an uncertain PDE constraint. In [100, 101, 175] stochastic gradient descent methods are developed for this problem class. We have already taken first steps to realize these methods in our framework and plan to employ the BMLMC method to further speed up the optimization inspired by the batched gradient descent. Lastly, we also take developments towards inverse UQ [217, 73, 74] into consideration. Solving Bayesian inverse problems requires an efficient evaluation of the forward model and thus, we suspect that we can contribute to this field with our integrated, budgeted and fully parallelized framework.

Acknowledgement

I gratefully acknowledges the support of the German Research Foundation (DFG) through an association with the CRC 1173. Furthermore, as the numerical experiments were executed on the HoReKa supercomputer, I highly appreciate the support and the chance to use this sophisticated piece of hardware funded by the Ministry of Science, Research and the Arts Baden-Württemberg and by the Federal Ministry of Education and Research. I also want personally thank all the people who have supported me during my PhD.

Persönliche Danksagung

Meine persönliche Motivation eine Promotion zu beginnen war es über mein Studium hinaus in einer akademischen Umgebung neue Inhalte zu erlernen und zu entwickeln. Hierfür hätte ich mir keinen besseren Ort als das IANM3 und den Lehrstuhl von Professor Wieners vorstellen können. Während meiner Zeit habe ich unglaublich viel über Mathematik - von Numerik bis zur Stochastik, über Softwareentwicklung - von C++ bis zu Automatisierung, und insbesondere über Teamarbeit und Kooperation gelernt. Diese Erfahrungen und dieses Wissen haben für mich einen unschätzbaren Wert und ich möchte allen die mich auf diesem Weg begleitet haben danken.

Herr Wieners, sie haben mir einen Großteil meines mathematischen Wissens während meines Studiums und während der Zeit als ihr Assistent vermittelt. Ich möchte mich insbesondere bei ihnen für ihr Vertrauen in meine Arbeit, die zahlreichen thematischen Diskussionen und die Freiheiten die sie mir gewährt haben bedanken. Darüber hinaus haben sie mich in die akademische Welt geführt und mir zu den richtigen Zeitpunkten neue Chancen ermöglicht. Ich bedanke mich bei ihnen für ihre Unterstützung und ihre Förderung und werde meine Promotion als eine äußerst spannende und prägende Zeit in Erinnerung behalten.

Professor Jahnke hat als Zweitgutachter zahlreiche Ungenauigkeiten und Fehler aufgedeckt. Tobias, ich weiß dein Feedback und deine Korrekturen sehr zu schätzen. Ich wusste wenn ich dich frage, dann bekomme ich eine ausführliche Korrektur und ein detailreiches Feedback. Ich bedanke mich für die Zeit die du in dieses Dokument investiert hast und hoffe, dass du etwas für dich daraus gewinnen konntest.

Außerdem möchte ich TT-Professor Sebastian Krumscheid danken. Auch wenn ich erst gegen Ende mit dir zusammen arbeiten durfte, so waren die fachlichen Diskussionen, deine Korrekturen und neuen Impulse entscheidend um die Arbeit abzurunden und abzuschließen. Danke Sebastian, ich hoffe auf zukünftige Zusammenarbeit.

Diese Arbeit definiert sich durch eine starke Softwarekomponente und Softwareentwicklung ist ohne jeden Zweifel ein Teamsport. Ich hatte das große Glück in einem unglaublich kompetenten und motivierten Team

zu arbeiten. Insbesondere möchte ich mich bei Daniele Corallo, Jonathan Wunderlich, Jonathan Fröhlich, Christian Rheinbay und Mathias Reichardt bedanken. Ich glaube wir haben zusammen große Arbeit geleistet und die Entwicklung von M++ neu gedacht. Für mich war, dank euch, die Entwicklung an M++ eine der besten Erfahrungen am IANM3. Daniele, du bist einer der hilfsbereitesten Menschen denen ich begegnet bin und die Kooperation mit dir am space-time Code war eine Achterbahnfahrt bei der ich mir keinen besseren Kollegen hätte vorstellen können. Jonathan W., ich danke dir für dein Zutun, deinen Blick für eine saubere Lösung und deine unzähligen Refactorings. Vieles wäre ohne dich nie umgesetzt worden. Insbesondere möchte ich auch Jonathan Fröhlich und Christian Rheinbay danken von denen ich viele Implementierungen verwendet habe. Ich habe sehr von eurer Arbeit profitiert und ohne eure Features wäre diese Doktorarbeit wesentlich schmaler ausgefallen. Außerdem möchte ich explizit unserem IT-Administrator Mathias Reichardt danken. Ohne dich Mathias, hätten wir keine unserer Ideen realisieren können. Ich bedanke mich für dein Engagement und den Hardware Support. M++ ist ein großes Softwareprojekt und ich will Johannes Ernesti, Laura Lindner, Laura Stengel, Daniel Ziegler, Lydia Wagner und Julian Krämer nicht unerwähnt lassen. Auch von deren Arbeit, sowie von unseren studentischen Hilfskräften David Schneiderhan, Tim Buchholz, Chinmay Patwardhan und Nico Fröhlich, habe ich profitieren dürfen.

Das Anfertigen einer Doktorarbeit ist ein langwieriger und fehleranfälliger Prozess. Deswegen will ich Benny Stein, Daniele Corallo, Jonathan Wunderlich, sowie meinen Freunden Stanislav Frolov und Robert Waddell fürs korrigieren danken. Ohne euer genaues Lesen, euren Rotstift und eure Hartnäckigkeit wären mir zahlreiche Ungenauigkeiten und Fehler nicht aufgefallen. Ich weiß euren Beitrag sehr zu schätzen.

Am IANM3 hatte ich zudem das große Glück stets Unterstützung in der Lehre oder am Tischkicker bei meinen restlichen Kollegen Daniel Weiß, Kevin Ganster, Johanna Mödl, Julian Baumstark, Michael Kirn, Lukas Pieronek, Zoltan Veszelka, Fabian Castelli und Pascal Kraft zu finden. Ich habe mich in diesem Kollegium sehr wohlgefühlt und viel gelernt.

Zuletzt bedanke ich mich bei meinem privaten Umfeld, bei meinen Eltern Lutz und Ulrike Baumgarten, bei meinen Schwestern Lina und Laura Baumgarten, meinem Freundeskreis und bei meiner Freundin Waranya Cusenza. ihr wart mir eine Stütze in den stressigen Phasen und ohne euch hätte jeglicher Ausgleich gefehlt. Danke, dass ihr für mich da wart und insbesondere danke Wara, dass du mich ertragen hast bei Frust und Überarbeitung. Die Menschen in meinem Umfeld mussten gegen Abschluss meiner Arbeit zurückstecken und ich weiß die Rücksichtnahme sehr zu schätzen.

A Appendix

A.1 Sobolev Spaces and Existence Theory

We shortly recall the definition of Sobolev spaces, weak solutions as well as some existence theory. We consider everything on a bounded and polygonal domain $D \subset \mathbb{R}^{\mathbf{d}}$ with the boundary ∂D.

Definition A.1. Let $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_k)^\top \in \mathbb{N}_0^k$ be a multiindex of order $k \in \mathbb{N}$ with $|\boldsymbol{\alpha}| := \alpha_1 + \dots + \alpha_k$ and $v, w \in \mathrm{L}^1_{\mathrm{loc}}(D)$ be locally integrable function. We call w the $\boldsymbol{\alpha}$-th *weak derivative* of v, if for all test functions $\phi \in C_0^\infty(D)$

$$\int_D v \mathrm{D}^{\boldsymbol{\alpha}} \phi \,\mathrm{d}\mathbf{x} = (-1)^{|\boldsymbol{\alpha}|} \int_D w \phi \,\mathrm{d}\mathbf{x} \qquad \text{with} \qquad \mathrm{D}^{\boldsymbol{\alpha}} \phi = \frac{\partial^{|\boldsymbol{\alpha}|} \phi}{\partial {\phi_1}^{\alpha_1} \dots \partial {\phi_k}^{\alpha_k}}.$$

By choosing the test function space to $C_0^\infty(D)$ the existence of $\mathrm{D}^{\boldsymbol{\alpha}} \phi$ is assured and the boundary terms vanish. The space of all functions with weak derivatives is a Sobolev space.

Definition A.2. Let $D \subset \mathbb{R}^{\mathbf{d}}$ and $p \in [1, \infty]$. The *Sobolev space* $\mathrm{W}^{k,p}(D)$ is the space of functions $v \in \mathrm{L}^p(D)$ for which all weak partial derivatives up to order k are contained in the Lebesgue space $\mathrm{L}^p(D)$. For functions

$v \in \mathrm{W}^{k,p}(D)$ the *Sobolev norm* is given by

$$\|v\|_{\mathrm{W}^{k,p}(D)} = \begin{cases} \left(\int_D \sum_{|\boldsymbol{\alpha}| \le k} |\mathrm{D}^{\boldsymbol{\alpha}} v|^p \,\mathrm{d}\mathbf{x} \right)^{\frac{1}{p}} , & p < \infty \\ \operatorname*{ess\,sup}_{\mathbf{x} \in \overline{D}} \sup_{|\boldsymbol{\alpha}| \le k} |\mathrm{D}^{\boldsymbol{\alpha}} v| , & p = \infty \end{cases} .$$

Definition A.3. There are some Sobolev spaces of particular interest for this work.

a) If $p = 2$, we write $\mathrm{H}^k(D) := \mathrm{W}^{k,2}(D)$. This is a Hilbert space with the inner product

$$\langle u, v \rangle_{\mathrm{H}^k(D)} = \sum_{|\boldsymbol{\alpha}| \le k} \langle \mathrm{D}^{\boldsymbol{\alpha}} u, \mathrm{D}^{\boldsymbol{\alpha}} v \rangle_{\mathrm{L}^2(D)} ,$$

and the induced (semi-)norm

$$\|v\|_{\mathrm{H}^k(D)} = \left(\int_D \sum_{|\boldsymbol{\alpha}| \le k} |\mathrm{D}^{\boldsymbol{\alpha}} v|^2 \,\mathrm{d}\mathbf{x} \right)^{\frac{1}{2}} \quad \text{and} \quad |v|_{\mathrm{H}^k(D)} = \left(\int_D \sum_{|\boldsymbol{\alpha}| = k} |\mathrm{D}^{\boldsymbol{\alpha}} v|^2 \,\mathrm{d}\mathbf{x} \right)^{\frac{1}{2}} .$$

b) The completion of $C_0^\infty(\overline{D})$ with respect to $\|\cdot\|_{\mathrm{H}^1(D)}$ is called $\mathrm{H}_0^1(D)$. The trace theorem [42, Chapter 3] verifies that this definition is meaningful and equivalent to

$$\mathrm{H}_0^1(D) = \left\{ v \in \mathrm{H}^1(D) : v|_{\partial D} = 0 \right\} .$$

Hence, $C_0^\infty(D)$ is dense in $\mathrm{H}_0^1(D)$.

c) The divergence Sobolev space is given by

$$\mathrm{H}(\mathrm{div}, D) := \left\{ \mathbf{v} \in \mathrm{L}^2(D, \mathbb{R}^{\mathbf{d}}) \colon \operatorname{div} \mathbf{v} \in \mathrm{L}^2(D) \right\} .$$

d) For any real $r \ge 0$ with $r \notin \mathbb{N}$ and $k \in \mathbb{N}$, we set $r = k + s$ with $0 < s < 1$. The *Sobolev–Slobodeckij* norm is defined by

$$\|v\|_{\mathrm{H}^r(D)} = \left(\|v\|_{\mathrm{H}^k(D)}^2 + |v|_{\mathrm{H}^r(D)}^2 \right)^{\frac{1}{2}} ,$$

where

$$|v|_{\mathrm{H}^r(D)} = \left(\int_{D \times D} \sum_{|\boldsymbol{\alpha}| = k} \frac{[\mathrm{D}^{\boldsymbol{\alpha}} v(\mathbf{x}_1) - \mathrm{D}^{\boldsymbol{\alpha}} v(\mathbf{x}_2)]^2}{\|\mathbf{x}_1 - \mathbf{x}_2\|^{d+2s}} \mathrm{d}\mathbf{x}_1 \mathrm{d}\mathbf{x}_2 \right)^{\frac{1}{2}} .$$

Then we can define the *broken* Sobolev space $\mathrm{H}^r(D)$ as the space of functions lying in $\mathrm{H}^k(D)$ for which the integral $|v|_{\mathrm{H}^r(D)}$ is finite.

Sobolev spaces can be used as solution spaces for the deterministic PDEs. To this end, we recall some existence theory.

Definition A.4. Let V be a Hilbert space with the inner product $\langle \cdot, \cdot \rangle$ and let $a\colon V \times V \to \mathbb{R}$ be a symmetric bilinear form, i.e., $a(u, v) = a(v, u)$ for all $u, v \in V$. The bilinear form is called *V-elliptic* if:

a) a is *continuous*, i.e., there exists $c_1 > 0$, such that $|a(u, v)| \leq c_1 \|u\| \|v\|$ for all $u, v \in V$.

b) a is *coercive*, i.e., there exists $c_2 > 0$, such that $a(v, v) \geq c_2 \|v\|^2$ for all $v \in V$.

Theorem A.5 (Lax-Milgram)**.** *Let V be a Hilbert space and $a\colon V \times V \to \mathbb{R}$ be a symmetric and V-elliptic bilinear form. Then, for every fixed and bounded $f \in V'$ the problem (2.8) has a unique solution $u_f \in V$ and we have $\|u_f\|_V \leq c_2^{-1} \|f\|_{V'}$*

Proof. The argument can for example be found in [84, Chapter 6.2]. □

Lastly, we give the inf-sup condition under which we can ensure well-posedness of problem 2.12. We follow the terminology of [81] and call this result Banach-Nečas-Babuška since it was formulated firstly by Nečas on Banach spaces and then popularized by Babuška in the context of FEM.

Theorem A.6 (Banach-Nečas-Babuška (BNB))**.** *Let exactly the setting of problem 2.12 be given, i.e., $(V, \|\cdot\|_V)$ and $(\tilde{V}, \|\cdot\|_{\tilde{V}})$ are Banach spaces with $\tilde{V}$ being reflexive. The problem is well-posed if and only if:*

$$\exists c > 0\colon \inf_{w \in V} \sup_{v \in \tilde{V}} \frac{a(w, v)}{\|w\|_V \|v\|_{\tilde{V}}} \geq c \quad \textit{and}$$
$$\forall v \in \tilde{V}\colon (\forall w \in V,\, a(w, v) = 0) \Rightarrow v = 0$$

The first condition is equivalent to

$$\forall w \in V\colon c \|w\|_V \leq \sup_{v \in \tilde{V}} \frac{a(w, v)}{\|v\|_{\tilde{V}}}.$$

Moreover, it holds for all $f \in V'$, that $\|u\|_V \leq c^{-1} \|f\|_{V'}$.

Proof. The proof is given in [81, Chapter 2, Appendix B], the a priori estimate can be directly concluded by

$$c \|u\|_V \leq \sup_{v \in \tilde{V}} \frac{a(u, v)}{\|v\|_{\tilde{V}}} = \sup_{v \in \tilde{V}} \frac{f(v)}{\|v\|_{\tilde{V}}} = \|f\|_{V'}.$$

□

A.2 Probability and Estimator Theory

We recall the majority of results from probability and estimator theory necessary for the investigations in this thesis. For a detailed introduction we refer to [169, 170] or [86].

Definition A.7. a) Let Ω be a set. The *power set* $2^\Omega := \{\mathcal{U} \colon \mathcal{U} \subseteq \Omega\}$ is the set of all subsets of Ω.

b) Let $\Omega \neq \emptyset$ be a set and 2^Ω be the power set. A system of sets $\mathcal{F} \subseteq 2^\Omega$ is called *σ-algebra*, if

$$\text{i) } \Omega \in \mathcal{F}, \qquad \text{ii) } \mathcal{B} \in \mathcal{F} \Rightarrow (\Omega \setminus \mathcal{B}) \in \mathcal{F}, \qquad \text{iii) } \{\mathcal{B}_m\}_{m\in\mathbb{N}} \subset \mathcal{F} \Rightarrow \bigcup_{m\in\mathbb{N}} \mathcal{B}_m \in \mathcal{F}.$$

c) Let Ω be a set and $\mathcal{F} \subseteq 2^\Omega$ be a σ-algebra. Let $\{\mathcal{B}_m\}_{m\in\mathbb{N}} \subset \mathcal{F}$ and $\mathcal{B}_m \cap \mathcal{B}_{m'} = \emptyset$ for all $m, m' \in \mathbb{N}$. A function $\mu\colon \mathcal{F} \to \mathbb{R}$ is called *measure*, if

$$\text{i) } \mu(\emptyset) = 0, \qquad \text{ii) } \forall \mathcal{B} \in \mathcal{F}\colon \mu(\mathcal{B}) \geq 0, \qquad \text{iii) } \mu\left(\bigcup_{m=1}^{\infty} \mathcal{B}_m\right) = \sum_{m=1}^{\infty} \mu(\mathcal{B}_m).$$

d) A function $\mathbb{P}\colon \mathcal{F} \to [0,1]$ is called *probability measure*, if $\mathbb{P}\colon \mathcal{F} \to [0,1]$ is measure with $\mathbb{P}(\Omega) = 1$.

Definition A.8. a) We call $(\Omega, \mathcal{F}, \mathbb{P})$ a *probability space*, if Ω is the space of random elementary events, $\mathcal{F} \subseteq 2^\Omega$ is a σ-algebra and $\mathbb{P}\colon \mathcal{F} \to [0,1]$ is the probability measure.

b) Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and $(\Xi, \mathcal{F}')$ be a measurable space. Then, a real valued *random variable* (*random vector*) is a measurable function $Y\colon \Omega \to \Xi \subset \mathbb{R}$ ($\mathbf{Y}\colon \Omega \to \Xi \subset \mathbb{R}^K$) mapping from the space of events into the space of possible outcomes.

c) Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and $Y\colon \Omega \to \Xi \subset \mathbb{R}$ be a random variable. Then, we call a fixed $\omega^{(m)} \in \Omega$ with $y^{(m)} = y(\omega^{(m)}) \in \Xi \subset \mathbb{R}$ a *realization* or a *sample*. We use m as superscript to distinguish between samples. Likewise for a random vector $\mathbf{Y}\colon \Omega \to \Xi \subset \mathbb{R}^K$, we call for a fixed $\omega^{(m)} \in \Omega$ with $\mathbf{y}^{(m)} = \mathbf{y}(\omega^{(m)}) \in \Xi \subset \mathbb{R}^K$ a *sample*.

At this point we remark, that most of this recall on probability theory is done for random variables and not random vectors in order to avoid an increased notational complexity. The intent is to fix on the terminology, however, everything defined below can be similarly defined for random vectors.

Definition A.9. a) For a given continuous random variable $Y\colon \Omega \to \Xi$, the *cumulative distribution function* (CDF) is given by

$$F_Y(y) := \mathbb{P}(Y \leq y) \in [0,1] \mathrel{\hat{=}} \text{probability that the random variable } Y \text{ is less than or equal to } y.$$

The CDF is a non-decreasing function $F_Y(y) \leq F_Y(y+\tilde{y})$ for some $\tilde{y} > 0$, hence, it can be used to quantify the probability that the random variable Y is between y_1 and y_2 by $\mathbb{P}(y_1 < Y \leq y_2) = F_Y(y_2) - F_Y(y_1)$.

b) If Y is a continuous random variable, the *probability density function* (PDF) is defined by

$$f_Y(y) = \frac{\mathrm{d}F_Y}{\mathrm{d}y} \quad \Rightarrow \quad \mathbb{P}(y_1 < Y \leq y_2) = \int_{y_1}^{y_2} f_Y(y)\mathrm{d}y.$$

c) If Y only has countably many discrete outcomes, we define the *probability mass function* (PMF)

$$f_Y(y) = \mathbb{P}(Y = y) \quad \Rightarrow \quad F_Y(y) = \sum_{Y \leq y} f_Y(y).$$

d) The joint CDF of two random variables Y_1, Y_2 is given by $F_{Y_1,Y_2}(y_1, y_2) := \mathbb{P}(Y_1 \le y_1 \wedge Y_2 \le y_2)$.

e) Let $\{Y_m\}_{m=1}^M$ be a sequence of random variables. We call these variables *identically distributed*, if they possess for all pairs $(m, \tilde{m})$ and all $y \in \Xi$ the same CDF $F_{Y_m}(y) = F_{Y_{\tilde{m}}}(y)$.

f) Let $\{Y_m\}_{m=1}^M$ be a sequence of random variables. We call these variables mutually *independent*, if $F_{Y_1,\dots,Y_M}(y_1,\dots,y_M) = \prod_{m=1}^M F_{Y_m}(y_m)$.

g) Let $\{Y_m\}_{m=1}^M$ be a sequence of random variables. If the random variables are independent and identically distributed, we call it *iid*.

h) Let Y_1 and Y_2 be two random variables with discrete outcome. The *conditional* PMF

$$\mathbb{P}(Y_1 = y_1 | Y_2 = y_2) = \frac{\mathbb{P}(Y_1 = y_1 \wedge Y_2 = y_2)}{\mathbb{P}(Y_2 = y_2)}, \quad \mathbb{P}(Y_2 = y_2) \neq 0$$

gives the probability to observe y_1 in Y_1, given that the outcome of Y_2 is y_2.

i) Let now Y_1 and Y_2 be continuous random variables. The *conditional* PDF is then

$$f_{Y_1|Y_2=y_2}(y_1) = \frac{f_{Y_1,Y_2}(y_1,y_2)}{f_{Y_2}(y_2)} \quad \text{with} \quad f_{Y_1,Y_2}(y_1,y_2) = \frac{\partial^2 F_{Y_1,Y_2}(y_1,y_2)}{\partial y_1 \partial y_2}.$$

Thereby this gives the PDF of Y_1, given that y_2 is observed in Y_2.

Theorem A.10 (Bayes' theorem)**.** *The conditional probability of a discrete random variable Y_1, given that Y_2 takes the value y_2 with $\mathbb{P}(Y_2 = y_2) \neq 0$, can be computed via*

$$\underbrace{\mathbb{P}(Y_1 = y_1 | Y_2 = y_2)}_{\hat{=}Posterior} = \frac{\mathbb{P}(Y_1 = y_1)\mathbb{P}(Y_2 = y_2 | Y_1 = y_1)}{\mathbb{P}(Y_2 = y_2)} \approx \underbrace{\mathbb{P}(Y_1 = y_1)}_{\hat{=}Prior} \underbrace{\mathbb{P}(Y_2 = y_2 | Y_1 = y_1)}_{\hat{=}Likelihood}.$$

Likewise, for continuous random variables it is

$$\underbrace{f_{Y_1|Y_2=y_2}(y_1)}_{\hat{=}Posterior} = \frac{f_{Y_1}(y_1) f_{Y_2|Y_1=y_1}(y_2)}{f_{Y_2}(y_2)} \approx \underbrace{f_{Y_1}(y_1)}_{\hat{=}Prior} \underbrace{f_{Y_2|Y_1=y_1}(y_2)}_{\hat{=}Likelihood}.$$

Proof. Simply follows by the definition of the conditional PMF (conditional PDF) in both directions. □

Bayes' theorem is of great importance in applied statistics and generally in the scientific method. It allows to update a *prior* believe (a hypothesis) about a random variable based on observed evidence. The *likelihood* models the probability of the evidence being true, given the hypothesis. The result of Bayes' theorem, commonly called *posterior*, is the probability of the hypothesis, given the observed evidence. If this probability is too small, we reject the hypothesis.

Definition A.11. For the following definitions, we assume that $Y : \Omega \to \Xi$ is sufficiently measurable, i.e., the integrals in the definitions below are finite.

a) The *median* is the point y, where the CDF takes the value $F_Y(y) = 0.5$, thereby the median splits all samples into equally sized halves.

b) The *expected value* (mean) μ_Y of a random variable $Y\colon \Omega \to \Xi$ is given by

$$\mu_Y := \mathbb{E}[Y] := \int_\Omega Y(\omega)\mathrm{d}\mathbb{P} = \int_{-\infty}^{\infty} y f_Y(y)\mathrm{d}y.$$

c) The *variance* σ_Y^2 of a random variable $Y\colon \Omega \to \Xi$ is given by

$$\sigma_Y^2 := \mathbb{V}[Y] := \mathbb{E}\left[(Y-\mu_Y)^2\right] = \mathbb{E}[Y^2] - 2\mu_Y\mathbb{E}[Y] + \mathbb{E}[\mu_Y^2] = \mathbb{E}[Y^2] - \mu_Y^2,$$

thereby it is the average of the squared distance between a random variable and its mean.

d) The *skewness* of a random variable $Y\colon \Omega \to \Xi$ is given by

$$\mathbb{S}[Y] := \mathbb{E}\left[\left(\frac{Y-\mu_Y}{\sigma_Y}\right)^3\right] = \frac{\mathbb{E}\left[(Y-\mu_Y)^3\right]}{\mathbb{V}[Y]^{3/2}},$$

which measures the symmetry of the distribution around the mean. A positive skewness means that the distribution leans to the left, where a negative skewness means that it leans to the right.

e) The *kurtosis* of a random variable, a measure of the tailedness of the PDF, is given by

$$\mathbb{K}[Y] := \mathbb{E}\left[\left(\frac{Y-\mu_Y}{\sigma_Y}\right)^4\right],$$

where a high kurtosis corresponds to greater outliers.

f) The q-th central moment is given by $\mathbb{M}_Y^q := \mathbb{E}[(Y-\mu_Y)^q]$ and we notice its use above.

Estimator theory. Estimators are rules to compute approximative statistical quantities of a random variable $Y\colon \Omega \to \Xi$. An estimator is depicted by equipping the random variable with a hat $\widehat{Y}$. As estimators are functions on random variables, they are random variables $\widehat{Y}\colon \Omega \to \Xi$ as well. Thereby, we can study everything from above equally well for estimators, say for example a *sample* of an estimator is an *estimate* which will be indexed with i to not confuse it with the sample (population) size M of the estimator. We recall a few properties of estimators (confer [86, 165]). Even though estimators can be used for other quantities as well, we will focus on estimating the expected value μ_Y.

Definition A.12. a) For a given estimate $\widehat{Y}^{(i)}$, the error is defined by $\mathrm{err}^{(i)} := \widehat{Y}^{(i)} - \mu_Y$.

b) An error independent of the individual estimate is the *(root) mean squared error* ((R)MSE)

$$\mathrm{err}_{\mathrm{MSE}}(\widehat{Y}) := \mathbb{E}\left[\left(\widehat{Y}-\mu_Y\right)^2\right], \quad \mathrm{err}_{\mathrm{RMSE}}(\widehat{Y}) := \left(\mathbb{E}\left[\left(\widehat{Y}-\mu_Y\right)^2\right]\right)^{1/2}.$$

It measures the squared distance, on average, of the estimates to the QoI.

c) Since estimators are random variables as well, we can define the *estimator variance*

$$\mathbb{V}[\widehat{Y}] := \mathbb{E}\left[\left(\widehat{Y} - \mathbb{E}[\widehat{Y}]\right)^2\right].$$

It measures the squared distance, on average, of the estimates to the expected estimates.

d) The *estimator bias* is given by $\mathbb{E}[\widehat{Y}] - \mu_Y$. It is called *unbiased*, if $\mathbb{E}[\widehat{Y}] = \mu_Y$.

e) Let the estimator be guided by the population size M. It is called *asymptotically unbiased*, if

$$\lim_{M\to\infty} \mathbb{E}[\widehat{Y}_M] = \mu_Y.$$

f) The *estimator tolerance* ϵ is given with the *confidence* $1-\delta$

$$\mathbb{P}\left(\left|\widehat{Y} - \mu_Y\right| > \epsilon\right) \leq \delta \quad\Longleftrightarrow\quad \mathbb{P}\left(\left|\widehat{Y} - \mu_Y\right| \leq \epsilon\right) > 1-\delta,$$

where $\delta \in (0,1)$ is a small probability.

g) Let the estimator be guided by the population size M. It is called *weakly consistent*, if for all $\epsilon > 0$

$$\lim_{M\to\infty} \mathbb{P}\left(\left|\widehat{Y}_M - \mu_Y\right| > \epsilon\right) = 0 \quad\Longleftrightarrow\quad \lim_{M\to\infty} \mathbb{P}\left(\left|\widehat{Y}_M - \mu_Y\right| \leq \epsilon\right) = 1$$

h) The estimator $\widehat{Y}_M$ is called *MSE-consistent*, if $\lim_{M\to\infty} \mathrm{err}_{\mathrm{MSE}} = 0$. Consistency can thereby be concluded by a vanishing estimator variance and asymptotic unbiasedness

$$\mathrm{err}_{\mathrm{MSE}}(\widehat{Y}) = \mathbb{E}\left[\left(\widehat{Y} - \mathbb{E}[\widehat{Y}] + \mathbb{E}[\widehat{Y}] - \mu_Y\right)^2\right] = \mathbb{V}[\widehat{Y}] + (\mathbb{E}[\widehat{Y}] - \mu_Y)^2 = \mathbb{V}[\widehat{Y}] + (\mathbb{E}[\widehat{Y} - \mu_Y])^2.$$

i) An estimator is called *efficient*, if out of all unbiased estimator it has the smallest variance.

In practice, an estimator $\widehat{Y}_1$ can only be called *more efficient* than another estimator $\widehat{Y}_2$. For biased estimators the MSE has to be considered to decide for the more efficient estimator instead of the variance. Furthermore, assuming that we can control the variance of the estimator $\mathbb{V}[\widehat{Y}]$ as well as the bias $(\mathbb{E}[\widehat{Y} - Y])^2$, we can formulate with $\theta \in (0,1)$ the bias-variance trade-off

$$\mathrm{err}_{\mathrm{MSE}}(\widehat{Y}) = \theta\mathbb{V}[\widehat{Y}] + (1-\theta)(\mathbb{E}[\widehat{Y} - \mu_Y])^2.$$

Bibliography

[1] Assyr Abdulle, Andrea Barth, and Christoph Schwab. “Multilevel Monte Carlo methods for stochastic elliptic multiscale PDEs”. In: *Multiscale Model. Simul.* 11.4 (2013), pp. 1033–1070. ISSN: 1540-3459. DOI: 10.1137/120894725. URL: https://doi.org/10.1137/120894725.

[2] R. Abgrall and S. Mishra. “Uncertainty quantification for hyperbolic systems of conservation laws”. In: *Handbook of numerical methods for hyperbolic problems.* Vol. 18. Handb. Numer. Anal. Elsevier/North-Holland, Amsterdam, 2017, pp. 507–544.

[3] Rémi Abgrall and Chi-Wang Shu, eds. *Handbook of numerical methods for hyperbolic problems.* Vol. 18. Handbook of Numerical Analysis. Applied and modern issues. Elsevier/North-Holland, Amsterdam, 2017, pp. xix+589. ISBN: 978-0-444-63910-3.

[4] P. Abrahamsen. *A review of Gaussian random fields and correlation functions.* Norwegian Computing Center, 1997.

[5] *ALSVID-UQ.* 2022. URL: https://www.sam.math.ethz.ch/alsvid-uq/.

[6] Hartwig Anzt et al. “An environment for sustainable research software in Germany and beyond - current state, open challenges, and call for action”. In: *F1000Research* 9 (2020).

[7] Hartwig Anzt et al. “Ginkgo: a modern linear operator algebra framework for high performance computing”. In: *ACM Trans. Math. Software* 48.1 (2022), Art. 2, 33. ISSN: 0098-3500. DOI: 10.1145/3480935. URL: https://doi.org/10.1145/3480935.

[8] Daniel Arndt et al. “The `deal.II` Library, Version 9.4”. In: *Journal of Numerical Mathematics* 30.3 (2022), pp. 231–246. DOI: 10.1515/jnma-2022-0054. URL: https://dealii.org/deal94-preprint.pdf.

[9] Douglas N. Arnold et al. “Unified analysis of discontinuous Galerkin methods for elliptic problems”. In: *SIAM J. Numer. Anal.* 39.5 (2001/02), pp. 1749–1779. ISSN: 0036-1429. DOI: 10.1137/S0036142901384162. URL: https://doi.org/10.1137/S0036142901384162.

[10] Søren Asmussen and Peter W. Glynn. *Stochastic simulation: algorithms and analysis.* Vol. 57. Stochastic Modelling and Applied Probability. Springer, New York, 2007, pp. xiv+476. ISBN: 978-0-387-30679-7.

[11] Quentin Ayoul-Guilmard et al. “Quantifying uncertain system outputs via the multi-level Monte Carlo method–distribution and robustness measures”. In: *arXiv preprint arXiv:2208.07252* (2022).

[12] Ivo Babuška, Fabio Nobile, and Raúl Tempone. “A stochastic collocation method for elliptic partial differential equations with random input data”. In: *SIAM J. Numer. Anal.* 45.3 (2007), pp. 1005–1034. ISSN: 0036-1429. DOI: 10.1137/050645142. URL: https://doi.org/10.1137/050645142.

[13] Ivo Babuška, Fabio Nobile, and Raúl Tempone. “A stochastic collocation method for elliptic partial differential equations with random input data”. In: *SIAM Rev.* 52.2 (2010), pp. 317–355. ISSN: 0036-1445. DOI: 10.1137/100786356. URL: https://doi.org/10.1137/100786356.

[14] Ivo Babuška, Raúl Tempone, and Georgios E. Zouraris. “Galerkin finite element approximations of stochastic elliptic partial differential equations”. In: *SIAM J. Numer. Anal.* 42.2 (2004), pp. 800–825. ISSN: 0036-1429. DOI: 10.1137/S0036142902418680. URL: https://doi.org/10.1137/S0036142902418680.

[15] Felix Bach et al. “The openCARP CDE–Concept for and implementation of a sustainable collaborative development environment for research software”. In: *arXiv preprint arXiv:2201.04434* (2022).

[16] Santiago Badia, Jerrad Hampton, and Javier Principe. “A Massively Parallel Implementation of Multilevel Monte Carlo for Finite Element Models”. In: *arXiv preprint arXiv -2111.11788* (2021).

[17] Santiago Badia, Jerrad Hampton, and Javier Principe. “Embedded multilevel Monte Carlo for uncertainty quantification in random domains”. In: *Int. J. Uncertain. Quantif.* 11.1 (2021), pp. 119–142. ISSN: 2152-5080. DOI: 10.1615/Int.J.UncertaintyQuantification.2021032984. URL: https://doi.org/10.1615/Int.J.UncertaintyQuantification.2021032984.

[18] Jayesh Badwaik et al. “Multilevel Monte Carlo finite volume methods for random conservation laws with discontinuous flux”. In: *ESAIM Math. Model. Numer. Anal.* 55.3 (2021), pp. 1039–1065. ISSN: 2822-7840. DOI: 10.1051/m2an/2021011. URL: https://doi.org/10.1051/m2an/2021011.

[19] Satish Balay et al. *PETSc Web page.* https://petsc.org/. 2022. URL: https://petsc.org/.

[20] Satish Balay et al. *PETSc/TAO Users Manual.* Tech. rep. ANL-21/39 - Revision 3.18. Argonne National Laboratory, 2022.

[21] Marco Ballesio et al. “Multilevel Monte Carlo acceleration of seismic wave propagation under uncertainty”. In: *GEM Int. J. Geomath.* 10.1 (2019), Paper No. 22, 43. ISSN: 1869-2672. DOI: 10.1007/s13137-019-0135-5. URL: https://doi.org/10.1007/s13137-019-0135-5.

[22] W. Bangerth and R. Rannacher. “Finite element approximation of the acoustic wave equation: error control and mesh adaptation”. In: *East-West J. Numer. Math.* 7.4 (1999), pp. 263–282. ISSN: 0928-0200.

[23] Pratyuksh Bansal et al. “Space-time discontinuous Galerkin approximation of acoustic waves with point singularities”. In: *IMA J. Numer. Anal.* 41.3 (2021), pp. 2056–2109. ISSN: 0272-4979. DOI: 10.1093/imanum/draa088. URL: https://doi.org/10.1093/imanum/draa088.

[24] Gabriel R Barrenechea et al. *Building bridges - connections and challenges in modern approaches to numerical partial differential equations.* Vol. 114. Springer, 2016.

[25] A. Barth, C. Schwab, and N. Zollinger. “Multilevel Monte Carlo Finite Element Method for elliptic PDEs with stochastic coefficients”. In: *Numerische Mathematik* 119.1 (2011), pp. 123–161.

[26] Len Bass, Paul Clements, and Rick Kazman. *Software architecture in practice.* Addison-Wesley Professional, 2003.

[27] Peter Bastian et al. “The Dune framework: basic concepts and recent developments”. In: *Comput. Math. Appl.* 81 (2021), pp. 75–112. ISSN: 0898-1221. DOI: 10.1016/j.camwa.2020.06.007. URL: https://doi.org/10.1016/j.camwa.2020.06.007.

[28] Niklas Baumgarten and Christian Wieners. “The parallel finite element system M++ with integrated multilevel preconditioning and multilevel Monte Carlo methods”. In: *Comput. Math. Appl.* 81 (2021), pp. 391–406. ISSN: 0898-1221. DOI: 10.1016/j.camwa.2020.03.004. URL: https://doi.org/10.1016/j.camwa.2020.03.004.

[29] Hamid Reza Bayat et al. “Numerical evaluation of discontinuous and nonconforming finite element methods in nonlinear solid mechanics”. In: *Comput. Mech.* 62.6 (2018), pp. 1413–1427. ISSN: 0178-7675. DOI: 10.1007/s00466-018-1571-z. URL: https://doi.org/10.1007/s00466-018-1571-z.

[30] Kent Beck et al. *The agile manifesto.* 2001. URL: https://agilemanifesto.org/.

[31] Richard Bellman. “Adaptive control processes: A guided tour”. In: (1961), pp. xvi+255.

[32] Richard Bellman. “Dynamic programming”. In: *Science* 153.3731 (1966), pp. 34–37.

[33] S. Ben Bader et al. “Space-time multilevel Monte Carlo methods and their application to cardiac electrophysiology”. In: *J. Comput. Phys.* 433 (2021), Paper No. 110164, 17. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2021.110164. URL: https://doi.org/10.1016/j.jcp.2021.110164.

[34] Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*. Vol. 1. Athena scientific, 2012.

[35] Marcel Bieri. "A sparse composite collocation finite element method for elliptic SPDEs". In: *SIAM J. Numer. Anal.* 49.6 (2011), pp. 2277–2301. ISSN: 0036-1429. DOI: 10.1137/090750743. URL: https://doi.org/10.1137/090750743.

[36] K. Binder and D. W. Heermann. "Monte Carlo simulation in statistical physics". In: Springer Series in Solid-State Sciences 80 (1992). An introduction, pp. viii+129. DOI: 10.1007/978-3-662-30273-6. URL: https://doi.org/10.1007/978-3-662-30273-6.

[37] Jacek Blazewicz et al. *Handbook on scheduling*. Springer, 2019.

[38] Daniele Boffi, Franco Brezzi, and Michel Fortin. *Mixed finite element methods and applications*. Vol. 44. Springer Series in Computational Mathematics. Springer, Heidelberg, 2013, pp. xiv+685. ISBN: 978-3-642-36518-8; 978-3-642-36519-5. DOI: 10.1007/978-3-642-36519-5. URL: https://doi.org/10.1007/978-3-642-36519-5.

[39] Thomas Bohlen et al. *Visco-acoustic full waveform inversion: from a DG forward solver to a Newton-CG inverse solver*. 2021. DOI: 10.1016/j.camwa.2021.09.001. URL: https://doi.org/10.1016/j.camwa.2021.09.001.

[40] Jan Bosch. "Continuous software engineering - An introduction". In: *Continuous software engineering*. Springer, 2014, pp. 3–13.

[41] George EP Box. "Robustness in the strategy of scientific model building". In: *Robustness in statistics*. Elsevier, 1979, pp. 201–236.

[42] Dietrich Braess. *Finite elements*. Third. Theory, fast solvers, and applications in elasticity theory, Translated from the German by Larry L. Schumaker. Cambridge University Press, Cambridge, 2007, pp. xviii+365. ISBN: 978-0-521-70518-9; 0-521-70518-5. DOI: 10.1017/CBO9780511618635. URL: https://doi.org/10.1017/CBO9780511618635.

[43] A. Brandt, M. Galun, and D. Ron. "Optimal multigrid algorithms for calculating thermodynamic limits". In: *Journal of Statistical Physics* 74.1 (1994), pp. 313–348.

[44] Susanne C. Brenner and L. Ridgway Scott. *The mathematical theory of finite element methods*. Third. Vol. 15. Texts in Applied Mathematics. Springer, New York, 2008, pp. xviii+397. ISBN: 978-0-387-75933-3. DOI: 10.1007/978-0-387-75934-0. URL: https://doi.org/10.1007/978-0-387-75934-0.

[45] Kurt M. Bretthauer and Bala Shetty. "The nonlinear knapsack problem—algorithms and applications". In: *European J. Oper. Res.* 138.3 (2002), pp. 459–472. ISSN: 0377-2217. DOI: 10.1016/S0377-2217(01)00179-5. URL: https://doi.org/10.1016/S0377-2217(01)00179-5.

[46] Franco Brezzi and Michel Fortin. *Mixed and hybrid finite element methods*. Vol. 15. Springer Science & Business Media, 2012.

[47] Alexander N. Brooks and Thomas J. R. Hughes. "Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations". In: *Comput. Methods Appl. Mech. Engrg.* 32.1-3 (1982). FENOMECH "81, Part I (Stuttgart, 1981), pp. 199–259. ISSN: 0045-7825. DOI: 10.1016/0045-7825(82)90071-8. URL: https://doi.org/10.1016/0045-7825(82)90071-8.

[48] Hans-Joachim Bungartz and Michael Griebel. "Sparse grids". In: *Acta numerica* 13.1 (2004), pp. 147–269.

[49] *bwUniCluster 2.0+GFB-HPC*. 2022. URL: https://www.scc.kit.edu/en/services/bwUniCluster_2.0.php.

[50] L. Caccetta and A. Kulanoot. “Computational aspects of hard knapsack problems”. In: *Nonlinear Anal.* 47.8 (2001), pp. 5547–5558. ISSN: 0362-546X. DOI: 10.1016/S0362-546X(01)00658-7. URL: https://doi.org/10.1016/S0362-546X(01)00658-7.

[51] Russel E. Caflisch. “Monte Carlo and quasi-Monte Carlo methods”. In: Acta Numer. 7 (1998), pp. 1–49. DOI: 10.1017/S0962492900002804. URL: https://doi.org/10.1017/S0962492900002804.

[52] Tony F. Chan, Gene H. Golub, and Randall J. LeVeque. “Algorithms for computing the sample variance: analysis and recommendations”. In: *Amer. Statist.* 37.3 (1983), pp. 242–247. ISSN: 0003-1305. DOI: 10.2307/2683386. URL: https://doi.org/10.2307/2683386.

[53] J. Charrier, R. Scheichl, and A. L. Teckentrup. “Finite element error analysis of elliptic PDEs with random coefficients and its application to multilevel Monte Carlo methods”. In: *SIAM J. Numer. Anal.* 51.1 (2013), pp. 322–352. ISSN: 0036-1429. DOI: 10.1137/110853054. URL: https://doi.org/10.1137/110853054.

[54] Julia Charrier. “Analyse numérique d’équations aux dérivées aléatoires, applications à l’hydrogéologie”. PhD thesis. École normale supérieure de Cachan-ENS Cachan, 2011.

[55] Julia Charrier. “Strong and weak error estimates for elliptic partial differential equations with random coefficients”. In: *SIAM J. Numer. Anal.* 50.1 (2012), pp. 216–246. ISSN: 0036-1429. DOI: 10.1137/100800531. URL: https://doi.org/10.1137/100800531.

[56] Jehanzeb H. Chaudhry, Nathanial Burch, and Donald Estep. “Efficient distribution estimation and uncertainty quantification for elliptic problems on domains with stochastic boundaries”. In: *SIAM/ASA J. Uncertain. Quantif.* 6.3 (2018), pp. 1127–1150. DOI: 10.1137/17M112230X. URL: https://doi.org/10.1137/17M112230X.

[57] Philippe G. Ciarlet. “Interpolation error estimates for the reduced Hsieh-Clough-Tocher triangle”. In: *Math. Comp.* 32.142 (1978), pp. 335–344. ISSN: 0025-5718. DOI: 10.2307/2006147. URL: https://doi.org/10.2307/2006147.

[58] K. A. Cliffe et al. “Multilevel Monte Carlo methods and applications to elliptic PDEs with random coefficients”. In: *Comput. Vis. Sci.* 14.1 (2011), pp. 3–15. ISSN: 1432-9360. DOI: 10.1007/s00791-011-0160-x. URL: https://doi.org/10.1007/s00791-011-0160-x.

[59] Bernardo Cockburn. “Discontinuous Galerkin methods”. In: *ZAMM Z. Angew. Math. Mech.* 83.11 (2003), pp. 731–754. ISSN: 0044-2267. DOI: 10.1002/zamm.200310088. URL: https://doi.org/10.1002/zamm.200310088.

[60] Bernardo Cockburn and Chi-Wang Shu. “Runge-Kutta discontinuous Galerkin methods for convection-dominated problems”. In: *J. Sci. Comput.* 16.3 (2001), pp. 173–261. ISSN: 0885-7474. DOI: 10.1023/A:1012873910884. URL: https://doi.org/10.1023/A:1012873910884.

[61] Nathan Collier et al. “A continuation multilevel Monte Carlo algorithm”. In: *BIT* 55.2 (2015), pp. 399–432. ISSN: 0006-3835. DOI: 10.1007/s10543-014-0511-3. URL: https://doi.org/10.1007/s10543-014-0511-3.

[62] Daniele Corallo, Willy Dörfler, and Christian Wieners. *Space-time discontinuous Galerkin methods for weak solutions of hyperbolic linear symmetric Friedrichs systems*. Tech. rep. 1. 2023, Paper No. 27, 30. DOI: 10.1007/s10915-022-02076-3. URL: https://doi.org/10.1007/s10915-022-02076-3.

[63] Thomas H Cormen et al. *Introduction to algorithms*. MIT press, 2022.

[64] R. Courant, K. Friedrichs, and H. Lewy. “Über die partiellen Differenzengleichungen der mathematischen Physik”. In: *Math. Ann.* 100.1 (1928), pp. 32–74. ISSN: 0025-5831. DOI: 10.1007/BF01448839. URL: https://doi.org/10.1007/BF01448839.

[65] Giuseppe Da Prato and Jerzy Zabczyk. *Stochastic equations in infinite dimensions.* Second. Vol. 152. Encyclopedia of Mathematics and its Applications. Cambridge University Press, Cambridge, 2014, pp. xviii+493. ISBN: 978-1-107-05584-1. DOI: 10.1017/CBO9781107295513. URL: https://doi.org/10.1017/CBO9781107295513.

[66] Constantine M. Dafermos. *Hyperbolic conservation laws in continuum physics.* Second. Vol. 325. Grundlehren der mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]. Springer-Verlag, Berlin, 2005, pp. xx+626. ISBN: 978-3-540-25452-2; 3-540-25452-8. DOI: 10.1007/3-540-29089-3. URL: https://doi.org/10.1007/3-540-29089-3.

[67] Gedeon Dagan. "Solute transport in heterogeneous porous formations". In: *Journal of fluid mechanics* 145 (1984), pp. 151–177.

[68] M. Dashti and A. M. Stuart. "Uncertainty quantification and weak approximation of an elliptic inverse problem". In: *SIAM J. Numer. Anal.* 49.6 (2011), pp. 2524–2542. ISSN: 0036-1429. DOI: 10.1137/100814664. URL: https://doi.org/10.1137/100814664.

[69] Michael Alan Howarth Dempster et al. *High-performance computing in finance: Problems, methods, and solutions.* CRC Press, 2018.

[70] Daniele Antonio Di Pietro and Alexandre Ern. *Mathematical aspects of discontinuous Galerkin methods.* Vol. 69. Mathématiques & Applications (Berlin) [Mathematics & Applications]. Springer, Heidelberg, 2012, pp. xviii+384. ISBN: 978-3-642-22979-4. DOI: 10.1007/978-3-642-22980-0. URL: https://doi.org/10.1007/978-3-642-22980-0.

[71] Josef Dick, Frances Y. Kuo, and Ian H. Sloan. "High-dimensional integration: the quasi-Monte Carlo way". In: *Acta Numer.* 22 (2013), pp. 133–288. ISSN: 0962-4929. DOI: 10.1017/S0962492913000044. URL: https://doi.org/10.1017/S0962492913000044.

[72] C. R. Dietrich and G. N. Newsam. "Fast and exact simulation of stationary Gaussian processes through circulant embedding of the covariance matrix". In: *SIAM J. Sci. Comput.* 18.4 (1997), pp. 1088–1107. ISSN: 1064-8275. DOI: 10.1137/S1064827592240555. URL: https://doi.org/10.1137/S1064827592240555.

[73] T. J. Dodwell et al. "A hierarchical multilevel Markov chain Monte Carlo algorithm with applications to uncertainty quantification in subsurface flow". In: *SIAM/ASA J. Uncertain. Quantif.* 3.1 (2015), pp. 1075–1108. DOI: 10.1137/130915005. URL: https://doi.org/10.1137/130915005.

[74] T. J. Dodwell et al. "Multilevel Markov chain Monte Carlo". In: *SIAM Rev.* 61.3 (2019). Revised reprint of "A hierarchical multilevel Markov chain Monte Carlo algorithm with applications to uncertainty quantification in subsurface flow", pp. 509–545. ISSN: 0036-1445. DOI: 10.1137/19M126966X. URL: https://doi.org/10.1137/19M126966X.

[75] Willy Dörfler, Stefan Findeisen, and Christian Wieners. "Space-time discontinuous Galerkin discretizations for linear first-order hyperbolic evolution systems". In: *Comput. Methods Appl. Math.* 16.3 (2016), pp. 409–428. ISSN: 1609-4840. DOI: 10.1515/cmam-2016-0015. URL: https://doi.org/10.1515/cmam-2016-0015.

[76] Willy Dörfler and Christian Wieners. "Space-time approximations for linear acoustic, elastic, and electro-magnetic wave equations". In: (2022).

[77] Willy Dörfler et al. "Parallel adaptive discontinuous Galerkin discretizations in space and time for linear elastic and acoustic waves". In: Radon Ser. Comput. Appl. Math. 25 ([2019] ©2019), pp. 61–88. DOI: 10.1515/9783110548488-002. URL: https://doi.org/10.1515/9783110548488-002.

[78] Stuart Dreyfus. "Richard Bellman on the birth of dynamic programming". In: *Oper. Res.* 50.1 (2002). 50th anniversary issue of Operations Research, pp. 48–51. ISSN: 0030-364X. DOI: 10.1287/opre.50.1.48.17791. URL: https://doi.org/10.1287/opre.50.1.48.17791.

[79] D. Drzisga et al. “Scheduling massively parallel multigrid for multilevel Monte Carlo methods”. In: *SIAM J. Sci. Comput.* 39.5 (2017), S873–S897. ISSN: 1064-8275. DOI: 10.1137/16M1083591. URL: https://doi.org/10.1137/16M1083591.

[80] Bo Einarsson, ed. *Accuracy and reliability in scientific computing.* Vol. 18. Software, Environments, and Tools. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2005, pp. xxiv+338. ISBN: 0-89871-584-9. DOI: 10.1137/1.9780898718157. URL: https://doi.org/10.1137/1.9780898718157.

[81] Alexandre Ern and Jean-Luc Guermond. *Theory and practice of finite elements.* Vol. 159. Applied Mathematical Sciences. Springer-Verlag, New York, 2004, pp. xiv+524. ISBN: 0-387-20574-8. DOI: 10.1007/978-1-4757-4355-5. URL: https://doi.org/10.1007/978-1-4757-4355-5.

[82] Johannes Ernesti. “Space-Time Methods for Acoustic Waves with Applications to Full Waveform Inversion”. PhD thesis. Karlsruher Institut für Technologie (KIT), 2018. 168 pp. DOI: 10.5445/IR/1000082807.

[83] Johannes Ernesti and Christian Wieners. “A space-time discontinuous Petrov-Galerkin method for acoustic waves”. In: Radon Ser. Comput. Appl. Math. 25 ([2019] ©2019), pp. 89–115. DOI: 10.1515/9783110548488-003. URL: https://doi.org/10.1515/9783110548488-003.

[84] Lawrence C. Evans. *Partial differential equations.* Second. Vol. 19. Graduate Studies in Mathematics. American Mathematical Society, Providence, RI, 2010, pp. xxii+749. ISBN: 978-0-8218-4974-3. DOI: 10.1090/gsm/019. URL: https://doi.org/10.1090/gsm/019.

[85] *ExaQUte XMC.* 2022. URL: https://gitlab.com/RiccardoRossi/exaqute-xmc.

[86] Ludwig Fahrmeir et al. *Statistik - Der Weg zur Datenanalyse.* Springer-Verlag, 2016.

[87] Zhiwei Fang et al. “Efficient stochastic Galerkin methods for Maxwell’s equations with random inputs”. In: *J. Sci. Comput.* 80.1 (2019), pp. 248–267. ISSN: 0885-7474. DOI: 10.1007/s10915-019-00936-z. URL: https://doi.org/10.1007/s10915-019-00936-z.

[88] Dave Farley. *Continuous Delivery Pipelines - How to Build Better Software Faster.* Pearson Education, 2020.

[89] Michael Feischl and Andrea Scaglioni. “Convergence of adaptive stochastic collocation with finite elements”. In: *Comput. Math. Appl.* 98 (2021), pp. 139–156. ISSN: 0898-1221. DOI: 10.1016/j.camwa.2021.07.001. URL: https://doi.org/10.1016/j.camwa.2021.07.001.

[90] Stefan Matthias Findeisen. “A Parallel and Adaptive Space-Time Method for Maxwell’s Equations”. PhD thesis. Karlsruher Institut für Technologie (KIT), 2016. 146 pp. DOI: 10.5445/IR/1000056876.

[91] *Finite Element Multiphysics PARallel solvers.* 2022. URL: https://github.com/fempar/fempar.

[92] Martin Frank, Jonas Kusch, and Jannick Wolters. “Entropy–Based Methods for Uncertainty Quantification of Hyperbolic Conservation Laws”. In: *Recent Advances in Numerical Methods for Hyperbolic PDE Systems.* Springer, 2021, pp. 29–56.

[93] Philipp Frauenfelder, Christoph Schwab, and Radu Alexandru Todor. “Finite elements for elliptic problems with stochastic coefficients”. In: *Comput. Methods Appl. Mech. Engrg.* 194.2-5 (2005), pp. 205–228. ISSN: 0045-7825. DOI: 10.1016/j.cma.2004.04.008. URL: https://doi.org/10.1016/j.cma.2004.04.008.

[94] Thomas Fritz et al. “Simulation of the contraction of the ventricles in a human heart model including atria and pericardium”. In: *Biomechanics and modeling in mechanobiology* 13.3 (2014), pp. 627–641.

[95] Johannes Fröhlich. “A segregated finite element method for cardiac elastodynamics in a fully coupled human heart model”. PhD thesis. Karlsruher Institut für Technologie (KIT), 2022.

[96] Edgar Gabriel et al. “Open MPI: Goals, concept, and design of a next generation MPI implementation”. In: *European Parallel Virtual Machine/Message Passing Interface Users’ Group Meeting*. Springer. 2004, pp. 97–104.

[97] Martin J. Gander. “50 years of time parallel time integration”. In: *Multiple shooting and time domain decomposition methods*. Vol. 9. Contrib. Math. Comput. Sci. Springer, Cham, 2015, pp. 69–113.

[98] M. Ganesh, Frances Y. Kuo, and Ian H. Sloan. “Quasi-Monte Carlo finite element analysis for wave propagation in heterogeneous random media”. In: *SIAM/ASA J. Uncertain. Quantif.* 9.1 (2021), pp. 106–134. DOI: 10.1137/20M1334164. URL: https://doi.org/10.1137/20M1334164.

[99] Robert N Gantner. “A generic C++ library for multilevel quasi-Monte Carlo”. In: *Proceedings of the Platform for Advanced Scientific Computing Conference*. 2016, pp. 1–12.

[100] Caroline Geiersbach and Georg Ch. Pflug. “Projected stochastic gradients for convex constrained problems in Hilbert spaces”. In: *SIAM J. Optim.* 29.3 (2019), pp. 2079–2099. ISSN: 1052-6234. DOI: 10.1137/18M1200208. URL: https://doi.org/10.1137/18M1200208.

[101] Caroline Geiersbach and Winnifried Wollner. “A stochastic gradient method with mesh refinement for PDE-constrained optimization under uncertainty”. In: *SIAM J. Sci. Comput.* 42.5 (2020), A2750–A2772. ISSN: 1064-8275. DOI: 10.1137/19M1263297. URL: https://doi.org/10.1137/19M1263297.

[102] Tobias Gerach et al. “Electro-mechanical whole-heart digital twins a fully coupled multi-physics approach”. In: *Mathematics* 9.11 (2021), p. 1247.

[103] Stephan Gerster and Michael Herty. “Entropies and symmetrization of hyperbolic stochastic Galerkin formulations”. In: *Commun. Comput. Phys.* 27.3 (2020), pp. 639–671. ISSN: 1815-2406. DOI: 10.4208/cicp.oa-2019-0047. URL: https://doi.org/10.4208/cicp.oa-2019-0047.

[104] Roger G Ghanem and Pol D Spanos. *Stochastic finite elements - a spectral approach*. Courier Corporation, 2003.

[105] A. D. Gilbert et al. “Analysis of quasi-Monte Carlo methods for elliptic eigenvalue problems with stochastic coefficients”. In: *Numer. Math.* 142.4 (2019), pp. 863–915. ISSN: 0029-599X. DOI: 10.1007/s00211-019-01046-6. URL: https://doi.org/10.1007/s00211-019-01046-6.

[106] M. B. Giles et al. “Quasi-Monte Carlo for finance applications”. In: *ANZIAM J.* 50.(C) (2008), pp. C308–C323. ISSN: 1446-1811.

[107] Michael Giles. *Multilevel Monte Carlo research worldwide*. 2022. URL: https://people.maths.ox.ac.uk/~gilesm/mlmc_community.html.

[108] Michael B. Giles. “Multilevel Monte Carlo methods”. In: *Acta Numer.* 24 (2015), pp. 259–328. ISSN: 0962-4929. DOI: 10.1017/S096249291500001X. URL: https://doi.org/10.1017/S096249291500001X.

[109] Michael B. Giles. “Multilevel Monte Carlo path simulation”. In: *Oper. Res.* 56.3 (2008), pp. 607–617. ISSN: 0030-364X. DOI: 10.1287/opre.1070.0496. URL: https://doi.org/10.1287/opre.1070.0496.

[110] Mike Giles. “Improved multilevel Monte Carlo convergence using the Milstein scheme”. In: (2008), pp. 343–358. DOI: 10.1007/978-3-540-74496-2_20. URL: https://doi.org/10.1007/978-3-540-74496-2_20.

[111] *GitLab CI/CD Documentation*. 2022. URL: https://docs.gitlab.com/ee/ci/.

[112] C. J. Gittelson. “Stochastic Galerkin discretization of the log-normal isotropic diffusion problem”. In: *Math. Models Methods Appl. Sci.* 20.2 (2010), pp. 237–263. ISSN: 0218-2025. DOI: 10.1142/S0218202510004210. URL: https://doi.org/10.1142/S0218202510004210.

[113] Claude Jeffrey Gittelson. “Adaptive Galerkin methods for parametric and stochastic operator equations”. PhD thesis. ETH Zurich, 2011.

[114] Paul Glasserman. "Monte Carlo methods in financial engineering". In: Applications of Mathematics (New York) 53 (2004). Stochastic Modelling and Applied Probability, pp. xiv+596.

[115] Edwige Godlewski and Pierre-Arnaud Raviart. *Numerical approximation of hyperbolic systems of conservation laws*. Vol. 118. Springer Science & Business Media, 2013.

[116] David Gottlieb and Dongbin Xiu. "Galerkin method for wave equations with uncertain coefficients". In: *Commun. Comput. Phys.* 3.2 (2008), pp. 505–518. ISSN: 1815-2406.

[117] I. G. Graham et al. "Analysis of circulant embedding methods for sampling stationary random fields". In: *SIAM J. Numer. Anal.* 56.3 (2018), pp. 1871–1895. ISSN: 0036-1429. DOI: 10.1137/17M1149730. URL: https://doi.org/10.1137/17M1149730.

[118] I. G. Graham et al. "Quasi-Monte Carlo methods for elliptic PDEs with random coefficients and applications". In: *J. Comput. Phys.* 230.10 (2011), pp. 3668–3694. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2011.01.023. URL: https://doi.org/10.1016/j.jcp.2011.01.023.

[119] Ivan G Graham, Matthew J Parkinson, and Robert Scheichl. "Full error analysis and uncertainty quantification for the heterogeneous transport equation in slab geometry". In: *arXiv preprint arXiv -1903.11838* (2019).

[120] Ivan G. Graham et al. "Circulant embedding with QMC: analysis for elliptic PDE with lognormal coefficients". In: *Numer. Math.* 140.2 (2018), pp. 479–511. ISSN: 0029-599X. DOI: 10.1007/s00211-018-0968-0. URL: https://doi.org/10.1007/s00211-018-0968-0.

[121] Richard L Graham, Timothy S Woodall, and Jeffrey M Squyres. "Open MPI: A flexible high performance massage passing interface". In: *International Conference on Parallel Processing and Applied Mathematics*. Springer. 2005, pp. 228–239.

[122] Marcus J. Grote, Simon Michel, and Fabio Nobile. "Uncertainty quantification by multilevel Monte Carlo and local time-stepping for wave propagation". In: *SIAM/ASA J. Uncertain. Quantif.* 10.4 (2022), pp. 1601–1628. DOI: 10.1137/21M1429047. URL: https://doi.org/10.1137/21M1429047.

[123] *Guidelines for Safeguarding Good Research Practice*. 2022. URL: https://www.dfg.de/en/research_funding/principles_dfg_funding/good_scientific_practice/.

[124] Philipp A. Guth et al. "A quasi-Monte Carlo method for optimal control under uncertainty". In: *SIAM/ASA J. Uncertain. Quantif.* 9.2 (2021), pp. 354–383. DOI: 10.1137/19M1294952. URL: https://doi.org/10.1137/19M1294952.

[125] Wolfgang Hackbusch. *Elliptic differential equations*. English. Vol. 18. Springer Series in Computational Mathematics. Theory and numerical treatment, Translated from the 1986 corrected German edition by Regine Fadiman and Patrick D. F. Ion. Springer-Verlag, Berlin, 2010, pp. xiv+311. ISBN: 978-3-642-05244-6. DOI: 10.1007/978-3-642-11490-8. URL: https://doi.org/10.1007/978-3-642-11490-8.

[126] Georg Hager and Gerhard Wellein. *Introduction to high performance computing for scientists and engineers*. CRC Press, 2010.

[127] Thomas Haigh, Mark Priestley, and Crispin Rope. "Los alamos bets on eniac: Nuclear monte carlo simulations, 1947-1948". In: *IEEE Annals of the History of Computing* 36.3 (2014), pp. 42–63.

[128] Abdul-Lateef Haji-Ali, Fabio Nobile, and Raúl Tempone. "Multi-index Monte Carlo: when sparsity meets sampling". In: *Numer. Math.* 132.4 (2016), pp. 767–806. ISSN: 0029-599X. DOI: 10.1007/s00211-015-0734-5. URL: https://doi.org/10.1007/s00211-015-0734-5.

[129] Abdul-Lateef Haji-Ali et al. "Multi-index stochastic collocation for random PDEs". In: *Comput. Methods Appl. Mech. Engrg.* 306 (2016), pp. 95–122. ISSN: 0045-7825. DOI: 10.1016/j.cma.2016.03.029. URL: https://doi.org/10.1016/j.cma.2016.03.029.

[130] Abdul-Lateef Haji-Ali et al. “Optimization of mesh hierarchies in multilevel Monte Carlo samplers”. In: *Stoch. Partial Differ. Equ. Anal. Comput.* 4.1 (2016), pp. 76–112. ISSN: 2194-0401. DOI: 10.1007/s40072-015-0049-7. URL: https://doi.org/10.1007/s40072-015-0049-7.

[131] Martin Hanke-Bourgeois. *Grundlagen der numerischen Mathematik und des wissenschaftlichen Rechnens.* Third. Vieweg + Teubner, Wiesbaden, 2009, p. 840. ISBN: 978-3-8348-0708-3. DOI: 10.1007/978-3-8348-9309-3. URL: https://doi.org/10.1007/978-3-8348-9309-3.

[132] Helmut Harbrecht, Michael Peters, and Markus Siebenmorgen. “Multilevel accelerated quadrature for PDEs with log-normally distributed diffusion coefficient”. In: *SIAM/ASA J. Uncertain. Quantif.* 4.1 (2016), pp. 520–551. DOI: 10.1137/130931953. URL: https://doi.org/10.1137/130931953.

[133] Helmut Harbrecht, Michael Peters, and Markus Siebenmorgen. “On multilevel quadrature for elliptic stochastic partial differential equations”. In: *Sparse grids and applications.* Vol. 88. Lect. Notes Comput. Sci. Eng. Springer, Heidelberg, 2013, pp. 161–179. DOI: 10.1007/978-3-642-31703-3. URL: https://doi.org/10.1007/978-3-642-31703-3.

[134] Ralf Hartmann. “Numerical analysis of higher order discontinuous Galerkin finite element methods”. In: (2008).

[135] S. Heinrich. “Multilevel Monte Carlo methods”. In: *LSSC* 1 (2001), pp. 58–67.

[136] Lukas Herrmann, Magdalena Keller, and Christoph Schwab. “Quasi-Monte Carlo Bayesian estimation under Besov priors in elliptic inverse problems”. In: *Math. Comp.* 90.330 (2021), pp. 1831–1860. ISSN: 0025-5718. DOI: 10.1090/mcom/3615. URL: https://doi.org/10.1090/mcom/3615.

[137] Nicholas J. Higham. *Accuracy and stability of numerical algorithms.* Second. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2002, pp. xxx+680. ISBN: 0-89871-521-0. DOI: 10.1137/1.9780898718027. URL: https://doi.org/10.1137/1.9780898718027.

[138] Viet H Hoang, Jia H Quek, and Christoph Schwab. “Multilevel MCMC Bayesian Inversion of Parabolic PDEs under Gaussian Prior”. In: *SAM Research Report* 2020 (2020).

[139] Viet Ha Hoang, Jia Hao Quek, and Christoph Schwab. “Analysis of a multilevel Markov chain Monte Carlo finite element method for Bayesian inversion of log-normal diffusions”. In: *Inverse Problems* 36.3 (2020), pp. 035021, 46. ISSN: 0266-5611. DOI: 10.1088/1361-6420/ab2a1e. URL: https://doi.org/10.1088/1361-6420/ab2a1e.

[140] Marlis Hochbruck and Alexander Ostermann. “Exponential integrators”. In: *Acta Numer.* 19 (2010), pp. 209–286. ISSN: 0962-4929. DOI: 10.1017/S0962492910000048. URL: https://doi.org/10.1017/S0962492910000048.

[141] Marlis Hochbruck et al. “Efficient time integration for discontinuous Galerkin approximations of linear wave equations [Plenary lecture presented at the 83rd Annual GAMM Conference, Darmstadt, 26th–30th March, 2012]”. In: *ZAMM Z. Angew. Math. Mech.* 95.3 (2015), pp. 237–259. ISSN: 0044-2267. DOI: 10.1002/zamm.201300306. URL: https://doi.org/10.1002/zamm.201300306.

[142] Robert J Hoeksema and Peter K Kitanidis. “Analysis of the spatial structure of properties of selected aquifers”. In: *Water resources research* 21.4 (1985), pp. 563–572.

[143] Paul Houston, Christoph Schwab, and Endre Süli. “Discontinuous *hp*-finite element methods for advection-diffusion-reaction problems”. In: *SIAM J. Numer. Anal.* 39.6 (2002), pp. 2133–2163. ISSN: 0036-1429. DOI: 10.1137/S0036142900374111. URL: https://doi.org/10.1137/S0036142900374111.

[144] Matteo Icardi, Gianluca Boccardo, and Raúl Tempone. “On the predictivity of pore-scale simulations. Estimating uncertainties with multilevel Monte Carlo”. In: *Advances in Water Resources* 95 (2016), pp. 46–60.

[145] Tobias Jahnke and Benny Stein. “A multilevel stochastic collocation method for Schrödinger equations with a random potential”. In: *SIAM/ASA J. Uncertain. Quantif.* 10.4 (2022), pp. 1753–1780. DOI: 10.1137/21M1440517. URL: https://doi.org/10.1137/21M1440517.

[146] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. "Reinforcement learning: A survey". In: *Journal of artificial intelligence research* 4 (1996), pp. 237–285.

[147] Jari Kaipio and Erkki Somersalo. *Statistical and computational inverse problems*. Vol. 160. Applied Mathematical Sciences. Springer-Verlag, New York, 2005, pp. xvi+339. ISBN: 0-387-22073-9.

[148] Peter Knabner and Lutz Angermann. "Numerical methods for elliptic and parabolic partial differential equations". In: Texts in Applied Mathematics 44 (2003), pp. xvi+424.

[149] Peter Knabner and Lutz Angermann. *Numerik partieller Differentialgleichungen. Eine anwendungsorientierte Einfuhrung*. 2000.

[150] Markus M. Knodel, Serge Kräutle, and Peter Knabner. "Global implicit solver for multiphase multicomponent flow in porous media with multiple gas components and general reactions: global implicit solver for multiple gas components". In: *Comput. Geosci.* 26.3 (2022), pp. 697–724. ISSN: 1420-0597. DOI: 10.1007/s10596-022-10140-y. URL: https://doi.org/10.1007/s10596-022-10140-y.

[151] Patrick Knupp and Kambiz Salari. *Verification of computer codes in computational science and engineering*. Discrete Mathematics and its Applications (Boca Raton). Chapman & Hall/CRC, Boca Raton, FL, 2003, pp. xiv+144. ISBN: 1-58488-264-6.

[152] Norbert Köckler. *Mehrgittermethoden - Ein Lehr-und Übungsbuch*. Springer-Verlag, 2012.

[153] Ralf Kornhuber, Christoph Schwab, and Maren-Wanda Wolf. "Multilevel Monte Carlo finite element methods for stochastic elliptic variational inequalities". In: *SIAM J. Numer. Anal.* 52.3 (2014), pp. 1243–1268. ISSN: 0036-1429. DOI: 10.1137/130916126. URL: https://doi.org/10.1137/130916126.

[154] Julian Gabriel Krämer. "A hybrid weakly conforming finite element method for applications in solid mechanics". PhD thesis. Karlsruher Institut für Technologie (KIT), 2020. 129 pp. DOI: 10.5445/IR/1000127572.

[155] S. Krumscheid, F. Nobile, and M. Pisaroni. "Quantifying uncertain system outputs via the multilevel Monte Carlo method—Part I: Central moment estimation". In: *J. Comput. Phys.* 414 (2020), pp. 109466, 21. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2020.109466. URL: https://doi.org/10.1016/j.jcp.2020.109466.

[156] Prashant Kumar et al. "A multigrid multilevel Monte Carlo method for transport in the Darcy-Stokes system". In: *J. Comput. Phys.* 371 (2018), pp. 382–408. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2018.05.046. URL: https://doi.org/10.1016/j.jcp.2018.05.046.

[157] Frances Y. Kuo, Christoph Schwab, and Ian H. Sloan. "Quasi-Monte Carlo finite element methods for a class of elliptic partial differential equations with random coefficients". In: *SIAM J. Numer. Anal.* 50.6 (2012), pp. 3351–3374. ISSN: 0036-1429. DOI: 10.1137/110845537. URL: https://doi.org/10.1137/110845537.

[158] Jonas Kusch. "Realizability-preserving discretization strategies for hyperbolic and kinetic equations with uncertainty". PhD thesis. Karlsruher Institut für Technologie (KIT), 2020.

[159] Pierre L'Ecuyer and François Panneton. "Fast random number generators based on linear recurrences modulo 2: overview and comparison". In: *Proceedings of the Winter Simulation Conference, 2005*. IEEE. 2005, 10–pp.

[160] J. Lang, R. Scheichl, and D. Silvester. "A fully adaptive multilevel stochastic collocation strategy for solving elliptic PDEs with random data". In: *J. Comput. Phys.* 419 (2020), pp. 109692, 17. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2020.109692. URL: https://doi.org/10.1016/j.jcp.2020.109692.

[161] Hans Petter Langtangen and Anders Logg. *Solving PDEs in python: the FEniCS tutorial I*. Springer Nature, 2017.

[162] Bernard Lapeyre et al. *Introduction to Monte Carlo methods for transport and diffusion equations.* Vol. 6. Oxford University Press on Demand, 2003.

[163] Sanghyun Lee, Young-Ju Lee, and Mary F. Wheeler. "A locally conservative enriched Galerkin approximation and efficient solver for elliptic and parabolic problems". In: *SIAM J. Sci. Comput.* 38.3 (2016), A1404–A1429. ISSN: 1064-8275. DOI: 10.1137/15M1041109. URL: https://doi.org/10.1137/15M1041109.

[164] Moritz Lenz, Moritz Lenz, and Anglin. *Python Continuous Integration and Delivery.* Springer, 2019.

[165] Fernando Puente León, Uwe Kiencke, and Fernando Puente León. *Messtechnik.* Vol. 10. Springer, 2015.

[166] RA LeVeque. *Numerical solution of hyperbolic conservation laws.* 2005.

[167] Alexander Litvinenko et al. "Computation of electromagnetic fields scattered from objects with uncertain shapes using multilevel Monte Carlo method". In: *IEEE Journal on Multiscale and Multiphysics Computational Techniques* 4 (2019), pp. 37–50.

[168] Jun S. Liu. *Monte Carlo strategies in scientific computing.* Springer Series in Statistics. Springer-Verlag, New York, 2001, pp. xvi+343. ISBN: 0-387-95230-6.

[169] Michel Loève. "Probability theory. I". In: Fourth. Graduate Texts in Mathematics, Vol. 45. Springer-Verlag, New York-Heidelberg, 1977, pp. xvii+425.

[170] Michel Loève. "Probability theory. II". In: Fourth. Graduate Texts in Mathematics, Vol. 46. Springer-Verlag, New York-Heidelberg, 1978, pp. xvi+413. ISBN: 0-387-90262-7.

[171] Yan Luo and Zhu Wang. "A multilevel Monte Carlo ensemble scheme for random parabolic PDEs". In: *SIAM J. Sci. Comput.* 41.1 (2019), A622–A642. ISSN: 1064-8275. DOI: 10.1137/18M1174635. URL: https://doi.org/10.1137/18M1174635.

[172] G. Malenova et al. "A sparse stochastic collocation technique for high-frequency wave propagation with uncertainty". In: *SIAM/ASA J. Uncertain. Quantif.* 4.1 (2016), pp. 1084–1110. DOI: 10.1137/15M1029230. URL: https://doi.org/10.1137/15M1029230.

[173] Axel Målqvist and Daniel Peterseim. "Localization of elliptic multiscale problems". In: *Math. Comp.* 83.290 (2014), pp. 2583–2603. ISSN: 0025-5718. DOI: 10.1090/S0025-5718-2014-02868-8. URL: https://doi.org/10.1090/S0025-5718-2014-02868-8.

[174] Stefano Marelli and Bruno Sudret. *UQLab: A framework for uncertainty quantification in Matlab.* American Society of Civil Engineers, 2014.

[175] Matthieu Martin, Sebastian Krumscheid, and Fabio Nobile. "Complexity analysis of stochastic gradient methods for PDE-constrained optimal control problems with uncertain parameters". In: *ESAIM Math. Model. Numer. Anal.* 55.4 (2021), pp. 1599–1633. ISSN: 2822-7840. DOI: 10.1051/m2an/2021025. URL: https://doi.org/10.1051/m2an/2021025.

[176] Daniel Maurer. "Ein hochskalierbarer paralleler direkter Löser für Finite Elemente Diskretisierungen". In: (2013).

[177] Daniel Maurer and Christian Wieners. "A parallel block LU decomposition method for distributed finite element matrices". In: *Parallel Computing* 37.12 (2011), pp. 742–758.

[178] Daniel Maurer and Christian Wieners. "A scalable parallel factorization of finite element matrices with distributed Schur complements". In: *Numer. Linear Algebra Appl.* 23.5 (2016), pp. 848–864. ISSN: 1070-5325. DOI: 10.1002/nla.2057. URL: https://doi.org/10.1002/nla.2057.

[179] Annamaria Mazzia. "Numerical methods for the solution of hyperbolic conservation laws". PhD thesis. Dipartimento di Metodi e Modelli Matematici per le Scienze Applicate, 1998.

[180] Ryan G. McClarren. *Uncertainty quantification and predictive computational science.* A foundation for physical scientists and engineers. Springer, Cham, 2018, pp. xvii+345. ISBN: 978-3-319-99524-3; 978-3-319-99525-0. DOI: 10.1007/978-3-319-99525-0. URL: https://doi.org/10.1007/978-3-319-99525-0.

[181] N. Metropolis. "The beginning of the Monte Carlo method". In: *Los Alamos Sci.* 15, Special Issue (1987). Stanislaw Ulam 1909–1984, pp. 125–130.

[182] Fabian Meyer, Christian Rohde, and Jan Giesselmann. "A posteriori error analysis for random scalar conservation laws using the stochastic Galerkin method". In: *IMA J. Numer. Anal.* 40.2 (2020), pp. 1094–1121. ISSN: 0272-4979. DOI: 10.1093/imanum/drz004. URL: https://doi.org/10.1093/imanum/drz004.

[183] S. Mishra and Ch. Schwab. "Sparse tensor multi-level Monte Carlo finite volume methods for hyperbolic conservation laws with random initial data". In: *Math. Comp.* 81.280 (2012), pp. 1979–2018. ISSN: 0025-5718. DOI: 10.1090/S0025-5718-2012-02574-9. URL: https://doi.org/10.1090/S0025-5718-2012-02574-9.

[184] S. Mishra, Ch. Schwab, and J. Šukys. "Multi-level Monte Carlo finite volume methods for nonlinear systems of conservation laws in multi-dimensions". In: *J. Comput. Phys.* 231.8 (2012), pp. 3365–3388. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2012.01.011. URL: https://doi.org/10.1016/j.jcp.2012.01.011.

[185] S. Mishra, Ch. Schwab, and J. Šukys. "Multi-level Monte Carlo finite volume methods for uncertainty quantification of acoustic wave propagation in random heterogeneous layered medium". In: *J. Comput. Phys.* 312 (2016), pp. 192–217. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2016.02.014. URL: https://doi.org/10.1016/j.jcp.2016.02.014.

[186] S. Mishra, Ch. Schwab, and J. Šukys. "Multilevel Monte Carlo finite volume methods for shallow water equations with uncertain topography in multi-dimensions". In: *SIAM J. Sci. Comput.* 34.6 (2012), B761–B784. ISSN: 1064-8275. DOI: 10.1137/110857295. URL: https://doi.org/10.1137/110857295.

[187] Siddhartha Mishra, Christoph Schwab, and Jonas Šukys. "Multi-level Monte Carlo finite volume methods for uncertainty quantification in nonlinear systems of balance laws". In: *Uncertainty quantification in computational fluid dynamics.* Vol. 92. Lect. Notes Comput. Sci. Eng. Springer, Heidelberg, 2013, pp. 225–294. DOI: 10.1007/978-3-319-00885-1_6. URL: https://doi.org/10.1007/978-3-319-00885-1_6.

[188] Siddhartha Mishra et al. "Numerical solution of scalar conservation laws with random flux functions". In: *SIAM/ASA J. Uncertain. Quantif.* 4.1 (2016), pp. 552–591. DOI: 10.1137/120896967. URL: https://doi.org/10.1137/120896967.

[189] Siddhartha Mishra et al. "Well-posedness of Bayesian inverse problems for hyperbolic conservation laws". In: *arXiv preprint arXiv:2107.09701* (2021).

[190] Mohammad Motamed, Fabio Nobile, and Raúl Tempone. "A stochastic collocation method for the second order wave equation with a discontinuous random speed". In: *Numer. Math.* 123.3 (2013), pp. 493–536. ISSN: 0029-599X. DOI: 10.1007/s00211-012-0493-5. URL: https://doi.org/10.1007/s00211-012-0493-5.

[191] Mohammad Motamed, Fabio Nobile, and Raúl Tempone. "Analysis and computation of the elastic wave equation with random coefficients". In: *Comput. Math. Appl.* 70.10 (2015), pp. 2454–2473. ISSN: 0898-1221. DOI: 10.1016/j.camwa.2015.09.013. URL: https://doi.org/10.1016/j.camwa.2015.09.013.

[192] Wolfgang Müller. "Numerische Analyse und parallele Simulation von nichtlinearen Cosserat-Modellen". PhD thesis. Karlsruhe Institute of Technology (KIT), 2009.

[193] ML Munoz-Ruiz, Carlos Pares, and Giovanni Russo. *Recent Advances in Numerical Methods for Hyperbolic PDE Systems.* Springer, 2021.

[194] *MUQ - The MIT Uncertainty Quantification Library*. 2022. URL: https://mituq.bitbucket.io/source/_site/index.html.

[195] RL Naff, DF Haley, and EA Sudicky. “High-resolution Monte Carlo simulation of flow and conservative transport in heterogeneous porous media - 1. Methodology and flow results”. In: *Water Resources Research* 34.4 (1998), pp. 663–677.

[196] RL Naff, DF Haley, and EA Sudicky. “High-resolution Monte Carlo simulation of flow and conservative transport in heterogeneous porous media - 2. Transport results”. In: *Water resources research* 34.4 (1998), pp. 679–697.

[197] *Netgen/NGSolve: high performance multiphysics finite element software*. 2022. URL: https://docu.ngsolve.org/latest/.

[198] Harald Niederreiter. “Quasi-Monte Carlo methods and pseudo-random numbers”. In: *Bull. Amer. Math. Soc.* 84.6 (1978), pp. 957–1041. ISSN: 0002-9904. DOI: 10.1090/S0002-9904-1978-14532-7. URL: https://doi.org/10.1090/S0002-9904-1978-14532-7.

[199] Harald Niederreiter. *Random number generation and quasi-Monte Carlo methods*. Vol. 63. CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1992, pp. vi+241. ISBN: 0-89871-295-5. DOI: 10.1137/1.9781611970081. URL: https://doi.org/10.1137/1.9781611970081.

[200] F. Nobile, R. Tempone, and C. G. Webster. “A sparse grid stochastic collocation method for partial differential equations with random input data”. In: *SIAM J. Numer. Anal.* 46.5 (2008), pp. 2309–2345. ISSN: 0036-1429. DOI: 10.1137/060663660. URL: https://doi.org/10.1137/060663660.

[201] F. Nobile, R. Tempone, and C. G. Webster. “An anisotropic sparse grid stochastic collocation method for partial differential equations with random input data”. In: *SIAM J. Numer. Anal.* 46.5 (2008), pp. 2411–2442. ISSN: 0036-1429. DOI: 10.1137/070680540. URL: https://doi.org/10.1137/070680540.

[202] F. Nobile and Raul Tempone. “Analysis and implementation issues for the numerical approximation of parabolic equations with random coefficients”. In: *Internat. J. Numer. Methods Engrg.* 80.6-7 (2009), pp. 979–1006. ISSN: 0029-5981. DOI: 10.1002/nme.2656. URL: https://doi.org/10.1002/nme.2656.

[203] Matthew Parno, Andrew Davis, and Linus Seelinger. “MUQ: The MIT Uncertainty Quantification Library”. In: *Journal of Open Source Software* 6.68 (2021), p. 3076. DOI: 10.21105/joss.03076. URL: https://doi.org/10.21105/joss.03076.

[204] Spassimir Paskov and Joseph F Traub. “Faster valuation of financial derivatives”. In: (1996).

[205] Nikhil Pathania. *Pro Continuous Delivery - With Jenkins 2.0*. Apress, 2017.

[206] Philippe Pébay et al. “Numerically stable, scalable formulas for parallel and online computation of higher-order multivariate central moments with arbitrary weights”. In: *Comput. Statist.* 31.4 (2016), pp. 1305–1325. ISSN: 0943-4062. DOI: 10.1007/s00180-015-0637-z. URL: https://doi.org/10.1007/s00180-015-0637-z.

[207] Ilaria Perugia, Christoph Schwab, and Marco Zank. “Exponential Convergence of *hp*-Time-Stepping in Space-Time Discretizations of Parabolic PDEs”. In: *arXiv preprint arXiv:2203.11879* (2022).

[208] M. Pisaroni, F. Nobile, and P. Leyland. “A continuation multi level Monte Carlo (C-MLMC) method for uncertainty quantification in compressible inviscid aerodynamics”. In: *Comput. Methods Appl. Mech. Engrg.* 326 (2017), pp. 20–50. ISSN: 0045-7825. DOI: 10.1016/j.cma.2017.07.030. URL: https://doi.org/10.1016/j.cma.2017.07.030.

[209] Robert Plato. *Numerische Mathematik kompakt*. Grundlagenwissen für Studium und Praxis. [Foundations for study and practice]. Friedr. Vieweg & Sohn, Braunschweig, 2000, pp. xiv+360. ISBN: 3-528-03153-0. DOI: 10.1007/978-3-322-96839-5. URL: https://doi.org/10.1007/978-3-322-96839-5.

[210] Gaël Poëtte, Bruno Després, and Didier Lucor. “Uncertainty quantification for systems of conservation laws”. In: *J. Comput. Phys.* 228.7 (2009), pp. 2443–2467. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2008.12.018. URL: https://doi.org/10.1016/j.jcp.2008.12.018.

[211] *PyMC*. 2022. URL: https://www.pymc.io/welcome.html.

[212] *RADAR4KIT, das Forschungsdaten Repositorium für das KIT*. 2022. URL: https://radar.kit.edu/.

[213] Sebastian Reiter et al. “A massively parallel geometric multigrid solver on hierarchically distributed grids”. In: *Computing and Visualization in Science* 16.4 (2013), pp. 151–164.

[214] Pieterjan Robbe, Dirk Nuyens, and Stefan Vandewalle. “A multi-index quasi–Monte Carlo algorithm for lognormal diffusion problems”. In: *SIAM J. Sci. Comput.* 39.5 (2017), S851–S872. ISSN: 1064-8275. DOI: 10.1137/16M1082561. URL: https://doi.org/10.1137/16M1082561.

[215] Christopher J. Roy and William L. Oberkampf. *A comprehensive framework for verification, validation, and uncertainty quantification in scientific computing*. Vol. 200. 25-28. 2011, pp. 2131–2144. DOI: 10.1016/j.cma.2011.03.016. URL: https://doi.org/10.1016/j.cma.2011.03.016.

[216] Martin Sauter. “Numerical Analysis of Algorithms for Infinitesimal Associated and Non-Associated Elasto-Plasticity”. PhD thesis. 2010. DOI: 10.5445/IR/1000019551.

[217] R. Scheichl, A. M. Stuart, and A. L. Teckentrup. “Quasi-Monte Carlo and multilevel Monte Carlo methods for computing posterior expectations in elliptic inverse problems”. In: *SIAM/ASA J. Uncertain. Quantif.* 5.1 (2017), pp. 493–518. DOI: 10.1137/16M1061692. URL: https://doi.org/10.1137/16M1061692.

[218] Volker Schmidt, ed. *Stochastic geometry, spatial statistics and random fields*. Vol. 2120. Lecture Notes in Mathematics. Models and algorithms, Selected papers from the Summer Academy on Stochastic Analysis, Modelling and Simulation of Complex Structures held in Hirschegg, September 11–24, 2011. Springer, Cham, 2015, pp. xxiv+464. ISBN: 978-3-319-10063-0; 978-3-319-10064-7.

[219] Andreas Schulz. “Numerical Analysis of the Electro-Magnetic Perfectly Matched Layer in a Discontinuous Galerkin Discretization”. PhD thesis. 2015. DOI: 10.5445/IR/1000047785.

[220] Katrin Schulz, Lydia Wagner, and Christian Wieners. “A mesoscale continuum approach of dislocation dynamics and the approximation by a Runge-Kutta discontinuous Galerkin method”. In: *International Journal of Plasticity* 120 (2019), pp. 248–261.

[221] Christoph Schwab, Ch Schwab, and CH Schwab. *p-and hp-finite element methods: Theory and applications in solid and fluid mechanics*. Oxford University Press, 1998.

[222] Christoph Schwab and Andreas Stein. “Multilevel Monte Carlo FEM for Elliptic PDEs with Besov Random Tree Priors”. In: *SAM Research Report* 2022 (2022).

[223] *SciPy quasi-Monte Carlo submodule*. URL: https://docs.scipy.org/doc/scipy/reference/stats.qmc.html#module-scipy.stats.qmc.

[224] Uta Seidler and Michael Griebel. “A Dimension-adaptive Combination Technique for Uncertainty Quantification”. In: *arXiv preprint arXiv:2204.05574* (2022).

[225] Ramin Shirazi Nejad. “A parallel elastic and inelastic heterogeneous multiscale method for rate-independent materials”. PhD thesis. Karlsruher Institut für Technologie (KIT), 2017. 120 pp. DOI: 10.5445/IR/1000074054.

[226] Ian H. Sloan and Henryk Woźniakowski. “When are quasi-Monte Carlo algorithms efficient for high-dimensional integrals?” In: *J. Complexity* 14.1 (1998), pp. 1–33. ISSN: 0885-064X. DOI: 10.1006/jcom.1997.0463. URL: https://doi.org/10.1006/jcom.1997.0463.

[227] Sergei Abramovich Smolyak. “Quadrature and interpolation formulas for tensor products of certain classes of functions”. In: *Doklady Akademii Nauk*. Vol. 148. 5. Russian Academy of Sciences. 1963, pp. 1042–1045.

[228] Il'ya Meerovich Sobol'. "On the distribution of points in a cube and the approximate evaluation of integrals". In: *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki* 7.4 (1967), pp. 784–802.

[229] Christian Soize. *Uncertainty quantification*. Springer, 2017.

[230] Andreas Stein and Andrea Barth. "A multilevel Monte Carlo algorithm for parabolic advection-diffusion problems with discontinuous coefficients". In: Springer Proc. Math. Stat. 324 ([2020] ©2020), pp. 445–466.

[231] Benny Stein. "Multi-level stochastic collocation methods for parabolic and Schrödinger equations". PhD thesis. Karlsruher Institut für Technologie (KIT), 2022.

[232] Peter Steinke. *Finite-Elemente-Methode*. Springer, 2015.

[233] Miroslav Stoyanov. "User manual: Tasmanian sparse grids v4. 0". In: *ORNL/TM-2015/596). Oak Ridge, TN: Oak Ridge National Laboratory* (2017).

[234] Miroslav Stoyanov et al. *Tasmanian*. Sept. 2013. DOI: 10.11578/dc.20171025.on.1087. URL: https%20-//github.com/ORNL/Tasmanian.

[235] Gilbert Strang. "Wavelets". In: *American Scientist* 82.3 (1994), pp. 250–255.

[236] A. M. Stuart. "Inverse problems: a Bayesian perspective". In: *Acta Numer.* 19 (2010), pp. 451–559. ISSN: 0962-4929. DOI: 10.1017/S0962492910000061. URL: https://doi.org/10.1017/S0962492910000061.

[237] Jonas Šukys. "Adaptive load balancing for massively parallel multi-level Monte Carlo solvers". In: *Parallel processing and applied mathematics. Part I.* Vol. 8384. Lecture Notes in Comput. Sci. Springer, Heidelberg, 2014, pp. 47–56. DOI: 10.1007/978-3-642-55224-3_5. URL: https://doi.org/10.1007/978-3-642-55224-3_5.

[238] Jonas Šukys. "Robust multi-level Monte Carlo Finite Volume methods for systems of hyperbolic conservation laws with random input data". PhD thesis. ETH Zurich, 2014.

[239] Jonas Šukys, Siddhartha Mishra, and Christoph Schwab. "Multi-level Monte Carlo finite difference and finite volume methods for stochastic linear hyperbolic systems". In: *Monte Carlo and quasi-Monte Carlo methods 2012*. Vol. 65. Springer Proc. Math. Stat. Springer, Heidelberg, 2013, pp. 649–666. DOI: 10.1007/978-3-642-41095-6_34. URL: https://doi.org/10.1007/978-3-642-41095-6_34.

[240] Jonas Šukys, Siddhartha Mishra, and Christoph Schwab. "Static load balancing for multi-level Monte Carlo finite volume solvers". In: *International Conference on Parallel Processing and Applied Mathematics*. Springer. 2011, pp. 245–254.

[241] T. J. Sullivan. *Introduction to uncertainty quantification*. Vol. 63. Texts in Applied Mathematics. Springer, Cham, 2015, pp. xii+342. ISBN: 978-3-319-23394-9; 978-3-319-23395-6. DOI: 10.1007/978-3-319-23395-6. URL: https://doi.org/10.1007/978-3-319-23395-6.

[242] Shuyu Sun and Jiangguo Liu. "A locally conservative finite element method based on piecewise constant enrichment of the continuous Galerkin method". In: *SIAM J. Sci. Comput.* 31.4 (2009), pp. 2528–2548. ISSN: 1064-8275. DOI: 10.1137/080722953. URL: https://doi.org/10.1137/080722953.

[243] Tao Tang and Tao Zhou. "Convergence analysis for stochastic collocation methods to scalar hyperbolic equations with a random wave speed". In: *Commun. Comput. Phys.* 8.1 (2010), pp. 226–248. ISSN: 1815-2406. DOI: 10.4208/cicp.060109.130110a. URL: https://doi.org/10.4208/cicp.060109.130110a.

[244] Phillip Taylor. "Simulating Gaussian Random Fields and Solving Stochastic Differential Equations using Bounded Wiener Increments". PhD thesis. University of Manchester, 2014.

[245] A. L. Teckentrup et al. "A multilevel stochastic collocation method for partial differential equations with random input data". In: *SIAM/ASA J. Uncertain. Quantif.* 3.1 (2015), pp. 1046–1074. DOI: 10.1137/140969002. URL: https://doi.org/10.1137/140969002.

[246] A. L. Teckentrup et al. “Further analysis of multilevel Monte Carlo methods for elliptic PDEs with random coefficients”. In: *Numer. Math.* 125.3 (2013), pp. 569–600. ISSN: 0029-599X. DOI: 10.1007/s00211-013-0546-4. URL: https://doi.org/10.1007/s00211-013-0546-4.

[247] Enric Tejedor et al. “PyCOMPSs: Parallel computational workflows in Python”. In: *The International Journal of High Performance Computing Applications* 31.1 (2017), pp. 66–82.

[248] Ekkachai Thawinan. “Numerical approximation of higher-dimensional Continuum Dislocation Dynamics theory in single crystal plasticity”. PhD thesis. 2015. DOI: 10.5445/IR/1000049994.

[249] *The HoreKa (Hochleistungsrechner Karlsruhe) supercomputer at KIT*. 2022. URL: https://www.nhr.kit.edu/userdocs/horeka/.

[250] *The Scalable Parallel Random Number Generators Library (SPRNG)*. 2022. URL: http://sprng.org/.

[251] *UM-Bridge*. 2022. URL: https://github.com/UM-Bridge/umbridge.

[252] *UQLab - The Framework for Uncertainty Quantification*. 2022. URL: https://www.uqlab.com/.

[253] Lydia Wagner. “A discontinuous Galerkin method for continuum dislocation dynamics in a fully-coupled elastoplasticity model”. PhD thesis. Karlsruher Institut für Technologie (KIT), 2019. 121 pp. DOI: 10.5445/IR/1000096183.

[254] Kerstin Weinberg and Christian Wieners. “Dynamic phase-field fracture with a first-order discontinuous Galerkin method for elastic waves”. In: *Comput. Methods Appl. Mech. Engrg.* 389 (2022), Paper No. 114330, 16. ISSN: 0045-7825. DOI: 10.1016/j.cma.2021.114330. URL: https://doi.org/10.1016/j.cma.2021.114330.

[255] B. P. Welford. “Note on a method for calculating corrected sums of squares and products”. In: *Technometrics* 4 (1962), pp. 419–420. ISSN: 0040-1706. DOI: 10.2307/1266577. URL: https://doi.org/10.2307/1266577.

[256] Christian Wieners. “A geometric data structure for parallel finite elements and the application to multigrid methods with block smoothing”. In: *Comput. Vis. Sci.* 13.4 (2010), pp. 161–175. ISSN: 1432-9360. DOI: 10.1007/s00791-010-0135-3. URL: https://doi.org/10.1007/s00791-010-0135-3.

[257] Christian Wieners. “Distributed point objects. A new concept for parallel finite elements”. In: *Domain decomposition methods in science and engineering*. Vol. 40. Lect. Notes Comput. Sci. Eng. Springer, Berlin, 2005, pp. 175–182. DOI: 10.1007/3-540-26825-1_14. URL: https://doi.org/10.1007/3-540-26825-1_14.

[258] Christian Wieners et al. *Mpp 3.1.1*. English. 2023. DOI: 10.35097/957.

[259] Christian Wieners et al. *Mpp 3.1.2*. English. 2023. DOI: 10.35097/958.

[260] Christian Wieners et al. *Mpp 3.1.3*. English. 2023. DOI: 10.35097/959.

[261] Jonathan Matthias Wunderlich. “Computer-assisted Existence Proofs for Navier-Stokes Equations on an Unbounded Strip with Obstacle”. PhD thesis. Karlsruher Institut für Technologie (KIT), 2022. 215 pp. DOI: 10.5445/IR/1000150609.

[262] Hans-Werner van Wyk. “Multilevel sparse grid methods for elliptic partial differential equations with random coefficients”. In: *arXiv preprint arXiv -1404.0963* (2014).

[263] Jiping Xin. “Boundary Element Approximation for Maxwell’s Eigenvalue Problem”. PhD thesis. 2011. DOI: 10.5445/IR/1000024374.

[264] Dongbin Xiu. “Numerical methods for stochastic computations”. In: (2010). A spectral method approach, pp. xiv+127.

[265] Dongbin Xiu. “Stochastic collocation methods: a survey”. In: (2017), pp. 699–716.

[266] Dongbin Xiu and Jie Shen. “Efficient stochastic Galerkin methods for random diffusion equations”. In: *J. Comput. Phys.* 228.2 (2009), pp. 266–281. issn: 0021-9991. doi: 10.1016/j.jcp.2008.09.008. url: https://doi.org/10.1016/j.jcp.2008.09.008.

[267] Adrian Zapletal et al. “The SoftWipe tool and benchmark for assessing coding standards adherence of scientific software”. In: *Scientific reports* 11.1 (2021), pp. 1–6.

[268] Dongxiao Zhang. *Stochastic methods for flow in porous media: coping with uncertainties*. Elsevier, 2001.

[269] Daniel Alexander Ziegler. “A parallel and adaptive space-time discontinuous Galerkin method for visco-elastic and visco-acoustic waves”. PhD thesis. Karlsruhe Institute of Technology (KIT), Nov. 2019. url: https%20-//doi.org/10.5445/IR/1000110469.

List of Figures

List of Tables

Code References

{1} https://git.scc.kit.edu/mpp/mpp/-/tags

{2} https://git.scc.kit.edu/mpp/mpp/

{3} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/src/lib2_mesh/mesh/Mesh.hpp

{4} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/src/lib2_mesh/meshparts/MeshPart.hpp

{5} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/tests/test2_mesh/mesh/TestMeshesCreator.hpp

{6} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/src/lib2_mesh/cells/Intval.hpp

{7} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/src/lib2_mesh/cells/Triangle.hpp

{8} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/src/lib2_mesh/cells/Quadrilateral.hpp

{9} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/src/lib2_mesh/cells/Hexahedron.hpp

{10} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/src/lib2_mesh/cells/TCell.hpp

{11} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/src/lib2_mesh/cells/TCell.hpp

{12} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/src/lib0_basic/parallel/Parallel.hpp

{13} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/src/lib4_fem/algebra/Vector.hpp

{14} https://git.scc.kit.edu/mpp/mpp

{15} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/src/lib2_mesh/mesh/CoarseGeometry.hpp

{16} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/src/lib3_disc/shapes

{17} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/src/lib3_disc/quadrature

{18} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/src/lib3_disc/dof

{19} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/src/lib4_fem/algebra/Vector.hpp

{20} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/src/lib4_fem/elements/ArgyrisElement.hpp

{21} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/src/lib4_fem/elements/CurlElement.hpp

{22} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/src/lib4_fem/elements/TaylorHoodElement.hpp

{23} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/spacetime/src/elements

{24} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/src/lib5_solve/solver/preconditioner/GaussSeidel.hpp

{25} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/src/lib5_solve/solver/preconditioner/Jacobi.hpp

{26} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/src/lib5_solve/solver/preconditioner/SuperLU.hpp

{27} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/src/lib5_solve/solver/preconditioner/Multigrid.hpp

{28} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/src/lib5_solve/solver/linearsolver/CG.hpp

{29} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/src/lib5_solve/solver/linearsolver/GMRES.hpp

{30} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/src/lib5_solve/solver/linearsolver/MINRES.hpp

{31} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/src/lib5_solve/solver/linearsolver/BiCGStab.hpp

{32} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/src/lib5_solve/timeintegrator/linear/RungeKutta.hpp

{33} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/src/lib5_solve/timeintegrator/linear/ExponentialIntegrator.hpp

{34} https://git.scc.kit.edu/mpp/mpp/-/blob/3.1.3/src/lib5_solve/spectrum

{35} https://git.scc.kit.edu/mpp/mluq/-/blob/1.2.0/mluq/src/generators/UniformDistribution.hpp

{36} https://git.scc.kit.edu/mpp/mluq/-/blob/1.2.0/mluq/src/generators/CirculantEmbedding.hpp

{37} https://git.scc.kit.edu/mpp/mluq/-/blob/1.2.0/mluq/src/estimators/datastructure/WelfordAggregate.hpp

{38} https://git.scc.kit.edu/mpp/mluq/-/blob/1.2.0/mluq/src/estimators/MultilevelEstimator.cpp

{39} https://git.scc.kit.edu/mpp/mluq/-/blob/1.2.0/mluq/src/estimators/MultilevelEstimator.cpp

{40} https://git.scc.kit.edu/mpp/mluq/-/blob/1.2.0/pipeline/mlmc-elliptic-on-horeka.yml

{41} https://git.scc.kit.edu/mpp/mluq/-/blob/1.2.0/pipeline/mlmc-transport-on-horeka.yml

{42} https://git.scc.kit.edu/mpp/mluq/-/blob/1.2.0/pipeline/mlmc-acoustic-time-stepping-on-horeka.yml

{43} https://git.scc.kit.edu/mpp/mluq/-/blob/1.2.0/pipeline/mlmc-acoustic-space-time-on-horeka.yml