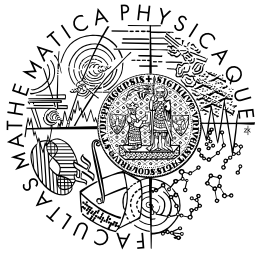


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Josef Martínek

Mixed Precision in Uncertainty Quantification Methods

Department of Numerical Mathematics

Supervisor of the master thesis: Dr. Erin Claire Carson, PhD.
Study programme: Mathematics
Study branch: Computational Mathematics

Prague 2023

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

I would like to express my gratitude to dr. Erin Carson and prof. Robert Scheichl for their help and valuable advice.

Title: Mixed Precision in Uncertainty Quantification Methods

Author: Josef Martínek

Department: Department of Numerical Mathematics

Supervisor: Dr. Erin Claire Carson, PhD., Department of Numerical Mathematics

Abstract: This work is concerned with analysing and exploiting mixed precision arithmetic in uncertainty quantification methods with emphasis on the multilevel Monte Carlo (MLMC) method. Although mixed precision can improve performance, it should be used carefully to avoid unwanted effects on the solution accuracy. We provide a rigorous analysis of uncertainty quantification methods in finite precision arithmetic. Based on this analysis, we exploit mixed precision arithmetic in uncertainty quantification methods to improve runtime while preserving the overall error.

We begin by stating the model problem, an elliptic PDE with random coefficients and a random right-hand side. Such a problem arises, for example, in uncertainty quantification for groundwater flow. Our focus is on approximating a quantity of interest given as the expected value of a functional of the solution of the PDE problem. To this end, we use the conforming finite element method for approximation in the spatial variable and the MLMC method for approximation of the expected value. We provide a novel rigorous analysis of the MLMC method in finite precision arithmetic and based on this we formulate an adaptive algorithm which determines the optimal precision value on each level of discretisation. To our knowledge, this is a new approach. Our theoretical results are then verified on numerous examples including an elliptic PDE with lognormal random coefficients, achieving a theoretical speedup of 4–8$\times$ compared to the reference double precision. The modeled speedup is achieved under the assumption that when single, half or quarter precision is used instead of double precision, the runtime is improved by the factor of 2, 4 or 8, respectively.

Keywords: Uncertainty quantification, Multilevel, Monte Carlo, Mixed precision, Iterative refinement

Contents

Introduction

Mathematical modelling has for a long time been an indispensable tool for understanding real world situations. Due to the fact that many parameters in the models are subject to uncertainty, the field of uncertainty quantification has recently gained considerable interest. One of the typical examples of this phenomenon arises in geosciences, namely, in the study of groundwater flow. A simple mathematical model for the flow is *Darcy's law*, which reads, in its primal form (involving the law of mass conservation; technical details omitted),

$$-\nabla \cdot \left(\frac{k}{\mu}\nabla p\right) = 0.$$

In this equation, the tensor k represents the permeability, a material parameter describing how easily water flows through the medium. The quantity μ is the dynamic viscosity of the fluid, and p is the pressure of the fluid; both are scalar. The uncertainty in this equation arises from the parameter k representing the permeability, which has to be obtained empirically in practice and is measured only in a few locations and then extrapolated to the entire domain; see [25], [22], [11], and the references therein. While our ideas can be straightforwardly generalised to different problems, we will be dealing with a *model problem* of the form

$$-\nabla \cdot \Big(a(\cdot,\omega)\nabla u(\cdot,\omega)\Big) = f(\cdot,\omega).$$

This elliptic PDE depending on a random parameter ω can be viewed as a generalisation of Darcy's law. We are interested in estimating the value of a quantity of interest (QOI), given as a function of the solution u. In this thesis we focus on the quantity of interest given by the expected value of a given functional of the solution, but other choices of the quantity of interest are possible in principle, such as the expected value of the value of the solution at a certain point of the domain. The coefficient a and the right-hand side f, depending on the random parameter ω, are assumed to be random fields, which are (possibly infinite-dimensional) random objects. If the random parameter ω is a high-dimensional random vector, the numerical approximation of the quantity of interest becomes exponentially difficult, which is known as the *curse of dimensionality*. Our model problem is a typical example of this (see [14], [12] and [23]), since the solution depends nonlinearly on the parameters, which increases the difficulty.

In order to approximate the quantity of interest, we have to first discretise the underlying PDE, since we are not able to solve the PDE analytically. To this end we use the finite element method (FEM), but any other suitable discretisation method can be used, as the discontinuous Galerkin method or the finite volume method, for example. The PDE discretisation introduces in our approximation the so-called discretisation error. After the PDE has been discretised, we have to face the challenge of approximating the integral in the expected value. To this end, the Monte Carlo (MC) methods are often used, which introduces in our approximation a second kind of error — the so-called sampling error. In the standard Monte Carlo method we need to take a sufficient number of samples of the discrete solution computed on a sufficiently fine mesh in order to bound the overall error. A significant variance reduction can be achieved if the samples

are taken on a hierarchy of discretisation levels. This is the underlying idea of the so-called *multilevel Monte Carlo* (MLMC); cf. [14], [6]. The MLMC method dramatically reduces the variance of the resulting estimator while preserving the overall cost and it allows us to balance the two errors — the discretisation error and the sampling error — efficiently.

In order to obtain the discrete PDE solution, a system of linear equations must be solved. This is when the third type of error comes into play, this being the algebraic error. The algebraic error stems from the fact that the system is solved on a computer using finite precision arithmetic. While much effort has been put into balancing the discretisation and sampling error, the impact of the algebraic error on the overall error has, to our knowledge, not been thoroughly studied. This is because the use of high-precision floating point arithmetic, e.g., double precision, has been standard practice in scientific computing for many years. In recent years, there has been growing interest in the selective use of low precision floating point arithmetic to accelerate scientific computations while maintaining acceptable levels of accuracy; cf. [20]. Recent advances in hardware and software have made such mixed precision algorithms a viable option for many scientific applications. Our intuition is that on coarser levels of discretisation, lower precision can safely be used without affecting the overall error. This intuition is confirmed by our rigorous analysis and leads to significant speedup of the computation.

To solve the resulting linear system we use a direct solver based on Cholesky factorization, although any other solver can be used in principle. We factorize the matrix at low precision, speeding up the computation significantly, and we iteratively improve our solution using iterative refinement until the desired accuracy is achieved. Iterative refinement is a well-known method for improving the accuracy of an approximate of a linear system; cf. [19], [27]. The first use of iterative refinement as well as the first rounding error analysis are attributed to Wilkinsion, who made use of the fact that inner products (and thus the residual) could be computed at twice the working precision on many computers of that period with no additional cost. The first error analysis in floating point arithmetic is attributed to Moler and can be viewed as a foundation for the recent analyses of iterative refinement. Today, iterative refinement is a widely used technique to get a better forward error of the solution, to recover the stability of the solver, and to accelerate the solution of linear system using low precision arithmetic; see [27] for an overview.

While Monte Carlo methods have been a focus of interest for many years, the use of lower precision arithmetic within them has not been rigorously analysed. The only work which goes in the direction of this thesis is the paper [4]. In this work, the authors explore the use of lower precision in the MLMC method on a model problem given by a stochastic differential equation discretised by the Euler method implemented on a field programmable gate array (FPGA). They propose a heuristic for choosing an appropriate precision on each level of discretisation based on testing the precisions one by one using 100 samples on each level. By precision they mean the number of bits in the significand (mantissa) of the floating point number. Using this approach, they achieve a speedup of 3—9$\times$ on their model problem on the same FPGA. We approach this topic from a different perspective and our results are original in many ways. Namely, we

- provide a rigorous analysis of Monte Carlo methods in finite precision arith-

metic via error estimates, namely of the standard MC method and the MLMC method (Sections 4.1 and 4.2.1);

- propose a novel adaptive mixed precision MLMC (MPMLMC) algorithm, determining the optimal precision on each level of discretisation using the theoretical error estimates with no additional cost (Section 4.2.4);
- provide the theoretical background for applying the adaptive algorithm to the model elliptic PDE with random coefficients and a random right-hand side (Section 4.2.2);
- extend the existing analysis of iterative refinement to linear systems coming from the finite element method (Section 3.3.1);
- demonstrate the efficiency of the MPMLMC algorithm through many numerical examples, typically achieving a theoretical speedup of 4–8× versus the standard approach (Chapter 5).

The adaptive MPMLMC algorithm can be straightforwardly employed when any direct or iterative solver is used for solving the underlying linear system. We choose to discuss in detail the case when a direct solver is used. Our results offer many options for generalisation and further research.

The thesis is divided into five chapters. The first chapter introduces the model problem given by the elliptic PDE in both the deterministic and the random case as well as its numerical solution by the finite element method. Selected convergence results of the FEM are presented and convergence of the FEM in finite precision arithmetic is discussed. In the second chapter we introduce the Monte Carlo methods, namely the standard MC method and the MLMC method. The corresponding complexity theorems for both methods are given as well as the adaptive MLMC algorithm. In the third chapter we provide the background of the numerical linear algebra methods used, including the basics of floating point arithmetic, Cholesky decomposition, and iterative refinement. Recent developments and possible gains from using the mixed-precision arithmetic are also overviewed. The fourth chapter contains the aforementioned analysis of Monte Carlo methods in finite precision along with the adaptive MPMLMC algorithm. Finally, the fifth chapter presents the numerical results along with a thorough discussion.

1. Finite element method

Our ultimate goal is to approximate a quantity of interest given in general as a function of the solution of a partial differential equation (PDE) with random input data. In order to approximate the quantity of interest, we have to first discretise the underlying PDE, since we are not able to solve the PDE analytically. To this end, we use the finite element method (FEM), but any other suitable discretisation method can be used, such as the discontinuous Galerkin method or the finite volume method, for example. This first chapter introduces in detail the model PDE problem and presents the basics of the finite element method. For simplicity, we start by formulating the deterministic problem. The generalization for the stochastic case is given in Section 1.3. The focus will be on the convergence results of the FEM needed in the subsequent chapters. In this chapter we follow the monograph [10].

1.1 Elliptic boundary value problem

As a representative example we will focus on an *elliptic boundary value problem* (BVP) defined as follows:

Problem 1.1 (Elliptic boundary value problem)**.** The problem of finding the solution of the equation

$$\begin{aligned} -\nabla \cdot (a \nabla u) &= f \quad \text{on } D, \\ u &= 0 \quad \text{on } \partial D, \end{aligned} \tag{1.1}$$

is called an *elliptic boundary value problem*. Here $D \subset \mathbb{R}^d$, $d \in \mathbb{N}$ is the *domain*, and we assume the boundary ∂D to be sufficiently smooth. Further $a : D \to \mathbb{R}^+$ is a *coefficient function* and $f : D \to \mathbb{R}$ is a *right-hand side*. The homogeneous Dirichlet boundary condition is employed and we assume the input data to be smooth, i.e., $f \in C(\bar{D})$ and $a \in C^1(\bar{D})$.

A function $u \in C^2(D) \cap C^1(\bar{D})$ is called a *classical solution* of the boundary value problem if it satisfies (1.1).

1.1.1 Weak formulation

Let us proceed by defining a weak formulation of Problem 1.1 in the standard way. We denote by $H^1(D)$ the *Sobolev space*,

$$H^1(D) \coloneqq \left\{ v \in L^2(D),\ \frac{\partial v}{\partial x_i} \in L^2(D)\ \forall i \in \{1, \dots, d\} \right\},$$

which is a Banach space with respect to the norm

$$\|v\|_{H^1(D)} \coloneqq \left(\int_D |v|^2 + \|\nabla v\|_{\mathbb{R}^d}^2 \mathrm{d}x \right)^{1/2}.$$

Here the partial derivatives are taken in the weak sense and the symbol $\|\cdot\|_{\mathbb{R}^d}$ denotes the Euclidean norm. Further we define $H_0^1(D) \coloneqq \overline{C_0^\infty(D)} \subset H^1(D)$,

where the completion is taken w.r.t. to the $\|\cdot\|_{H^1(D)}$ norm. The space H_0^1 is a Hilbert space with respect to the inner product

$$(u,v)_{H_0^1(D)} \coloneqq (\nabla u, \nabla v)_{L^2(D)}, \quad u,v \in H_0^1(D),$$

The associated norm on $H_0^1(D)$ is given by $|v|_{H^1(D)} = \|v\|_{H_0^1(D)} \coloneqq \sqrt{(v,v)_{H_0^1(D)}}$.

Having introduced the necessary notions, we can proceed to the weak formulation:

Problem 1.2 (Abstract variational problem)**.** Let $V \coloneqq H_0^1(D)$ and define $\tilde{a} : V \times V \to \mathbb{R}$ and $l : V \to \mathbb{R}$ by

$$\tilde{a}(u,v) \coloneqq \int_D a(x)\nabla u(x) \cdot \nabla v(x) \mathrm{d}x \quad \text{and} \quad l(v) \coloneqq \int_D f(x)v(x)\mathrm{d}x,$$

respectively, for fixed a and f from (1.1). The problem of finding $u \in V$ such that

$$\tilde{a}(u,v) = l(v) \quad \forall v \in V$$

is called the weak formulation of the elliptic boundary value problem (Problem 1.1). It will also be referred to as the *abstract variational problem (AVP).*

For the purpose of the weak formulation it is sufficient to assume that $f \in L^2(D)$ and $a \in L^\infty(D)$. Formally we can obtain the weak formulation by multiplying (1.1) by a test function $v \in V$, integrating over the domain D, and using Green's theorem along with the boundary condition. To be able to prove the existence and uniqueness of the solution of this problem, we have to impose an additional condition on the input data.

Assumption 1.3. The coefficient function a from Problem 1.2 satisfies that there exist $a_{min}, a_{max} \in \mathbb{R}$ such that for almost every (a.e.) $x \in D$, we have

$$0 < a_{min} \leq a(x) \leq a_{max} < \infty.$$

To solve the abstract variational problem (Problem 1.2), we can now use the Lax-Milgram lemma:

Lemma 1.4 (Lax-Milgram)**.** *Let V be a real Hilbert space equipped with the norm $\|\cdot\|_V$. Let $\tilde{a} : V \times V \to \mathbb{R}$ be a coercive (with a constant α) and bounded bilinear form, i.e.,*

$$\begin{aligned} \tilde{a}(v,v) &\geq \alpha \|v\|_V^2 \quad \forall v \in V, \\ |\tilde{a}(u,v)| &\leq c\|u\|_V \|v\|_V \quad \forall u,v \in V, \end{aligned}$$

and let $l : V \to \mathbb{R}$ be a bounded linear functional. Then there exists a unique $u \in V$ such that for all $v \in V$ it holds that $\tilde{a}(u,v) = l(v)$ and we have the a-priori estimate

$$\|u\|_V \leq \frac{1}{\alpha}\|l\|_{V'},$$

where V' denotes the dual space of the space V.

To show the existence and uniqueness of the solution of Problem 1.2 it is now sufficient to verify the assumptions of the Lax-Milgram lemma. To this end, we set $V = H_0^1(D)$ and $\|\cdot\|_V = |\cdot|_{H_0^1(D)}$. The boundedness and coercivity of the bilinear form $\tilde{a}$ follow from Assumption 1.3. The boundedness of the linear functional l in the $H_0^1(D)$ norm follows from the Poincaré-Friedrichs inequality. The Poincaré-Friedrichs inequality reads that for the bounded domain D there exists $C > 0$ such that for all $v \in H_0^1(D)$ we have

$$\|v\|_{L^2(D)} \leq C\|v\|_{H_0^1(D)}.$$

By this, the assumptions of the Lax-Milgram lemma are verified and the well-posedness of the solution of Problem 1.2 follows. See Chapter 3 in [10] for details.

1.2 Finite element discretisation of the BVP

One of the most prominent and general schemes for finding an approximate solution to the abstract variational problem (Problem 1.2) is the so-called *Galerkin method*. This method proceeds by defining a finite-dimensional subspace $V_n \subset V$ and then solving the restricted problem: Find $u_n \in V_n$ such that

$$a(u_n, v_n) = l(v_n) \quad \text{for all } v_n \in V_n. \tag{1.2}$$

The finite element method is then a Galerkin-type method given by a special choice of the subspace V_n which we will describe now. For the rest of the thesis we are going to assume that $D \subset \mathbb{R}^2$ and that D is a polygon. In short, we can say that throughout this thesis we will be using the conforming finite element method with a shape-regular and quasi-uniform system of triangulations. The triangulations will consist of triangle-shaped elements. The space V_n will consist of continuous piecewise-linear functions on D, i.e., functions which are affine on every triangle and continuous across the triangle edges. The precise meaning of these words will be given now.

1.2.1 Domain discretisation

Definition (Finite element)**.** *A* finite element *is a triple* (K, P_K, Ψ_K) *such that*

1. $K \subset \mathbb{R}^2$ *is a triangle with vertices* y_1*,* y_2*, and* y_3 *and it is a closed set,*
2. P_K *is the space of all affine functions on* K*, and*
3. Ψ_K *is given by*

$$\Psi_K = \{\psi_j : P_K \to \mathbb{R},\ \psi_j(p) = p(y_j),\ j = 1, 2, 3\}.$$

For brevity, we sometimes speak about "an element K*" instead of "an element* (K, P_K, Ψ_K)*".*

Definition (Triangulation τ_h)**.** *A triangulation (mesh)* τ_h *of the domain* D *is defined as a partition of the domain* D*,* $\tau_h := \{K_i;\ i = 1, \ldots, N\}$*, where* K_i *is an element (in the sense of the definition above) for every* $i \in \{1, \ldots, N\}$*. A triangulation* τ_h *is assumed to satisfy the following assumptions:*

1. $\overline{D} = \bigcup_{K \in \tau_h} K$ *and*

2. *for each* K_1, $K_2 \in \tau_h$, *exactly one of the following is true:*

   (a) $K_1 \cap K_2 = \emptyset$,

   (b) $K_1 \cap K_2$ *is a common vertex or a common edge.*

Further we define a discretisation parameter h *of the triangulation* τ_h *by*

$$h := \max_{K \in \tau_h} diam\, K.$$

Given a system of triangulations $\{\tau_h\}$, we assume its *shape-regularity*. Let us denote by ρ_K the radius of the inscribed circle in K and $h_K := \operatorname{diam} K$. The system of triangulations is said to be shape-regular if

$$\exists C > 0\, \forall \tau_h \in \{\tau_h\}\, \forall K \in \tau_h : \frac{\rho_K}{h_K} > C.$$

Moreover, we assume the *quasi-uniformity* of the family of meshes. A family of meshes $\{\tau_h\}$ is said to be quasi-uniform if it is shape-regular and

$$\exists c > 0\, \forall \tau_h \in \{\tau_h\}\, \forall K \in \tau_h : h_K > ch.$$

Let us note that the definition of triangulation can be generalised in terms of both the domain and the finite elements. For instance, it is possible to consider $D \subset \mathbb{R}^n$, $n > 2$, where D has a more diverse shape than polygonal/polyhedral. Also the definition of the finite element can be widely generalised to a more complex shape K, a more general space P_K, and also a more general set of the degrees of freedom Ψ_K. Please refer to [10] for the general definition. However, to illustrate the principles of the use of mixed precision arithmetic in uncertainty quantification methods, we are going to use only this simple setting. A more general version of the FEM can be readily employed in practical problems.

1.2.2 Finite element space

We proceed to the definition of the finite element space V_h. As suggested above, we will be using the FE space of piecewise linear functions. Let $\mathcal{P}_1(K)$ denote the space of polynomials of total degree 1 on K, where $K \in \tau_h$ for a given triangulation τ_h. We then define V_h by

$$V_h := \{v \in C(\overline{D}) : v|_K \in \mathcal{P}_1(K) \text{ for all } K \in \tau_h\} \subset H^1(D).$$

Moreover we define the space $V_{h,0} \subset V_h$ which corresponds to the space V_n from the Galerkin method; cf. (1.2). It is defined as

$$V_{h,0} := \{v \in V_h : v|_{\partial D} = 0\} \subset H_0^1(D). \tag{1.3}$$

The function values on the boundary are considered in the sense of traces. After Problem 1.2 has been discretised using the FEM, the discrete problem has to be solved. The discrete problem leads to a system of linear algebraic equations which has to be solved numerically. To this end, we use some of the numerical linear algebra methods described in Chapter 3. Having introduced the preliminaries, we now describe for completeness the discrete abstract variational problem.

1.2.3 Discrete abstract variational problem

The FEM proceeds by choosing $V_n \coloneqq V_{h,0}$ in the Galerkin method (1.2). The discrete problem then reads

Problem 1.5 (Discrete abstract variational problem)**.** Find $u_h \in V_{h,0}$ such that

$$\tilde{a}(u_h, v_h) = l(v_h) \quad \forall v \in V_{h,0},$$

where $\tilde{a}$ and l are defined as in the abstract variational problem (Problem 1.2).

The unique solvability of the discrete abstract variational problem (Problem 1.5) follows from the Lax-Milgram lemma (Lemma 1.4), since $V_{h,0} \subset V = H_0^1(D)$ and thus the assumptions of the Lax-Milgram lemma are satisfied similarly to the AVP 1.2.

As mentioned above, the discrete problem leads to a system of linear algebraic equations. Since solving this system is an important part of this thesis, we will now focus on the derivation of this system in more detail.

Consider a triangulation τ_h of the domain D. Let us denote by n the number of vertices of the triangulation which do not belong to ∂D, i.e.,

$$n \coloneqq \operatorname{card}\Big(\{y_i,\, y_i \text{ is a vertex of } K,\, K \in \tau_h,\, y_i \notin \partial D\}\Big).$$

Further let $\Phi \coloneqq \{\phi_1, \ldots, \phi_n\}$ denote the basis of $V_{h,0}$ such that

$$\begin{aligned} \phi_j(y_i) &= \delta_{i,j}, \\ \operatorname{supp} \phi_j &= \bigcup_{\substack{y_j \in K, \\ K \in \tau_h}} K, \end{aligned} \tag{1.4}$$

where $i, j = 1, \ldots, n$.

Let us now express the finite element solution $u_h \in V_{h,0}$ of Problem 1.5 in terms of the basis Φ, i.e.,

$$u_h = \sum_{j=1}^{n} x_j \phi_j.$$

Due to the linearity of the bilinear form $\tilde{a}$ and the functional l, the discrete AVP (Problem 1.5) can be equivalently reformulated as

$$\sum_{j=1}^{n} x_j \tilde{a}(\phi_j, \phi_i) = l(\phi_i), \quad i = 1, \ldots, n, \tag{1.5}$$

which can be rewritten as a system of linear algebraic equations. Define

$$b_i \coloneqq l(\phi_i) \quad \text{and} \quad a_{i,j} \coloneqq \tilde{a}(\phi_j, \phi_i).$$

We obtain what will be referred to as the *FE system*

$$Ax = b, \tag{1.6}$$

where $A = (a_{i,j})_{i,j=1}^{n} \in \mathbb{R}^{n \times n}$ is the *stiffness matrix*, $b = (b_i)_{i=1}^{n} \in \mathbb{R}^n$ is the right-hand side and $x = (x_i)_{i=1}^{n} \in \mathbb{R}^n$ is the vector of unknowns. Solving this system is equivalent to solving Problem 1.5.

From the fact that the bilinear form $\tilde{a}$ is symmetric and coercive it follows that the matrix A is symmetric and positive definite. Note also that because of the property (1.4) the matrix A is sparse. All these properties are referred to later when the numerical solution of the system is discussed.

1.2.4 Convergence results of the FEM

This section presents convergence results of the FEM needed in the subsequent chapters. We do not aim to provide thorough analysis of the convergence results, but a brief overview of the results important for our purposes.

In order to obtain the desired estimates, we need our solution u to be of higher regularity. Therefore we need to impose, on top of Assumption 1.3, one additional assumption on the AVP 1.2:

Assumption 1.6. Consider the abstract variational problem (Problem 1.2). The domain D is assumed to be convex and coefficient function a is assumed to be Lipschitz continuous on D.

Lemma 1.7 (FEM convergence in the L^2 norm)**.** *Let $\{\tau_h\}$ be a system of triangulations of the domain D. Let $u_h \in V_{h,0}$ be the solution of the discrete abstract variational problem (Problem 1.5) corresponding to the triangulation τ_h. Then the following estimate holds:*

$$\|u - u_h\|_{L^2(D)} \leq Ch^2 \|f\|_{L^2(D)},$$

where $C > 0$ is independent of h and f.

Proof. In a standard way, via the Aubin-Nitsche duality trick, we obtain

$$\|u - u_h\|_{L^2(D)} \leq Ch^2 |u|_{H^2(D)}; \tag{1.7}$$

see [10], Section 2.3.4. Here C is independent of h and u and $|\cdot|_{H^2(D)}$ denotes the H^2-seminorm. As proved in [26], Lemma 5.1 and Lemma 5.2, Assumption 1.6 ensures that the solution u is H^2-regular, i.e., $u \in H^2(D)$ and

$$\exists C > 0 \, \forall f \in L^2(D) : \; |u|_{H^2(D)} \leq C \|f\|_{L^2(D)}. \tag{1.8}$$

Note that [26] deals with the problem with random data. The proof in the deterministic case can be found in [15], Section 5.2. The statement of the lemma is now an immediate consequence of the inequalities (1.7) and (1.8). □

The equation (1.8) proving the H^2-regularity of the solution is a standard result from PDE theory in the case when we assume convexity of the domain; see the monograph [15].

1.2.5 FEM in finite precision arithmetic

Until now we have considered the underlying PDE problem and its solution via the finite element method. We have considered the FE discretisation and consequently, the exact solution u_h of the discrete problem. However, in practice the solution u_h is not obtained exactly due to the limitations of finite precision arithmetic, rather we compute an approximation $\widehat{u}_h$ of the discrete solution u_h. The aim of this section is to estimate the error $\|u_h - \widehat{u}_h\|_{H_0^1(D)}$ by means of the residual of the solution of the FE system $Ax = b$. Let us first introduce the notation.

Let $\widehat{x} \in \mathbb{R}^n$ be an approximate solution of the FE system (1.6). The approximation (reconstruction) $\widehat{u}_h$ of the FE solution u_h is then given by

$$\widehat{u}_h = \sum_{j=1}^{n} \widehat{x}_j \phi_j.$$

Recall that the FE solution u_h is given by $u_h = \sum_{j=1}^{n} x_j \phi_j$, where $x \in \mathbb{R}^n$ is the solution of the FE system $Ax = b$. Let us denote by r the residual of the approximate solution $\widehat{x}$, i.e., $r \coloneqq A\widehat{x} - b$. The following lemma provides us with an estimate of $\|u_h - \widehat{u}_h\|_{H_0^1(D)}$ using the residual r. It is a corollary of Proposition 9.19 from [10].

Lemma 1.8. *Let u_h be the solution of the discrete AVP (Problem 1.5) and let $\widehat{u}_h$ be the approximation of u_h defined above. Then*

$$\|u_h - \widehat{u}_h\|_{H_0^1(D)} \leq C \|f\|_{L^2(D)} \frac{\|r\|_{\mathbb{R}^n}}{\|b\|_{\mathbb{R}^n}},$$

where C is independent of h and u and f. The symbol $\|\cdot\|_{\mathbb{R}^n}$ denotes the Euclidean norm.

Proof. Let us verify the assumptions of Proposition 9.19 from [10]. The proposition assumes that the so-called discrete inf-sup condition holds; see (9.8) in the aforementioned book. The discrete inf-sup condition is equivalent to well-posedness of the discrete problem (Problem 1.5) as stated below Remark 2.20 in [10], which means that we require the unique solvability of the problem and the a-priori estimate for the norm of the solution. The discrete problem (Problem 1.5) is well-posed owing to the Lax-Milgram lemma and thus the discrete inf-sup condition holds. The next assumption reads (transcribed to our notation)

$$\exists g \in L^2(D)\, \forall v_h \in V_{h,0} : |l(v)| \leq \|g\|_{L^2(D)} \|v_h\|_{L^2(D)}.$$

This follows immediately from the definition of l (see Problem 1.2) and Hölder's inequality. By this the assumptions of the theorem are verified. The fact that C is independent of h follows from the discussion under Proposition 9.19 in [10], since we assume the family of triangulations $\{\tau_h\}$ to be quasi-uniform. Note that for the condition number of the mass matrix $\mathcal{M}_t$ (see Proposition 9.19 in [10]) to be independent of the discretisation parameter h regardless of the dimension of the domain, only the quasi-uniformity of the mesh is required, see Section 9.1.3, Theorem 9.8 in [10]. Also our bilinear form $\tilde{a}_h$ is uniformly bounded with respect to h due to the fact that in our case $\tilde{a}_h = \tilde{a}$ (i.e., the bilinear form itself is not discretized in any way). □

1.3 Finite element method for problems with random data

Until now we have considered only the standard finite element method for problems that do not include any uncertainties. However, in the problems which we encounter, uncertainty plays a crucial role. We therefore introduce uncertainty

in our model elliptic boundary value problem and then discuss the finite element method applied to this problem. This will be used in the uncertainty quantification algorithms described in Chapter 2. More details can be found in [2] and [26].

1.3.1 Elliptic BVP with random data

We begin with the theoretical preliminaries. In the following definition we introduce the so-called *random field*. The goal here is to introduce uncertainty in the coefficient and the right-hand side of the boundary value problem (Problem 1.1). The definition generalises in some sense the coefficient a and the right-hand side f from the BVP and therefore we use the same notation.

Definition 1.9. *Let $D \subset \mathbb{R}^2$ be the domain as in the BVP (Problem 1.1). Let $(\Omega, \mathcal{U}, \mathbb{P})$ be a probability space, where Ω is an abstract set of elementary events, $\mathcal{U}$ is the σ-algebra of events and $\mathbb{P} : \mathcal{U} \to [0,1]$ is a probability measure. A* random field *is a mapping*

$$a : D \times \Omega \to \mathbb{R}$$

satisfying that for all $x \in D$ the function $a(x, \cdot) : \Omega \to \mathbb{R}$ is a random variable.

Definition. *For a fixed $\omega \in \Omega$ the function $a(\cdot, \omega) : D \to \mathbb{R}$ is called a* realization *of the random field a.*

We often need to work with the norm of a random field. A common approach is to define abstract Bochner spaces, a generalization of Lebesgue spaces, and measure the norm of the random field in a suitable Bochner space, as is done in [26], for example. We choose not to introduce here the fully general definition of Bochner spaces. Instead, we introduce only the necessary notation so that we can work with random fields comfortably:

Definition. *Let $p, q \in [1, \infty)$. Let f be a random field such that for a.e. $\omega \in \Omega$ it holds that $f(\cdot, \omega) \in L^p(D)$. Then we define*

$$\|f\|_{L^q(\Omega, L^p(D))} = \left(\int_\Omega \|f(\cdot, \omega)\|^q_{L^p(D)} d\omega \right)^{1/q},$$

if the integral on the right-hand side converges.

Let us now state the elliptic boundary value problem with random data which is a direct generalization of the BVP (Problem 1.1).

Problem 1.10 (BVP with random data)**.** Let $(\Omega, \mathcal{U}, \mathbb{P})$ be a probability space and $\omega \in \Omega$. The problem of finding a solution to the equation

$$\begin{aligned} -\nabla \cdot \Big(a(\cdot, \omega) \nabla u(\cdot, \omega) \Big) &= f(\cdot, \omega) \quad \text{on } D, \\ u(\cdot, \omega) &= 0 \quad \text{on } \partial D, \end{aligned} \tag{1.9}$$

is called the *boundary value problem with random data*. Here we assume that the random fields f and a satisfy $f(\cdot, \omega) \in C(\overline{D})$ and $a(\cdot, \omega) \in C^1(\overline{D})$.

Let us note that after a $\omega \in \Omega$ has been sampled, the equation (1.9) for this fixed ω does not contain any randomness.

In the same way we now generalise the abstract variational problem (Problem 1.2):

Problem 1.11 (AVP with random data)**.** Let $(\Omega, \mathcal{U}, \mathbb{P})$ be a probability space and $\omega \in \Omega$. Recall that $V = H_0^1(D)$ and let us define $\tilde{a} : V \times V \times \Omega \to \mathbb{R}$ and $l : V \times \Omega \to \mathbb{R}$ by

$$\tilde{a}\Big(u(\cdot,\omega), v, \omega\Big) \coloneqq \int_D a(x,\omega)\nabla u(x,\omega) \cdot \nabla v(x) \mathrm{d}x \quad \text{and}$$
$$l(v,\omega) \coloneqq \int_D f(x,\omega) v(x) \mathrm{d}x,$$

respectively, for fixed random fields a and f satisfying $a(\cdot,\omega) \in L^\infty(D)$ and $f(\cdot,\omega) \in L^2(D)$.

If $u(\cdot,\omega) \in V$ satisfies

$$\tilde{a}\Big(u(\cdot,\omega), v, \omega\Big) = l(v,\omega) \quad \forall v \in V$$

for a.e. $\omega \in \Omega$, then u is called a weak solution of the elliptic boundary value problem with random data (Problem 1.10). The problem of finding such a u will also be referred to as the *abstract variational problem with random data.*

In order to prove the unique solvability of the AVP with random data, we impose one additional assumption, analogous to Assumption 1.3 in the deterministic case.

Assumption 1.12. The random field a from the AVP with random data (Problem 1.11) satisfies that there exist a_{min} and a_{max} such that for a.e. $\omega \in \Omega$ and a.e. $x \in D$ it holds that

$$0 < a_{min} \leq a(x,\omega) \leq a_{max} < \infty.$$

Let us remark that this assumption can be somewhat weakened by making the coefficients a_{min} and a_{max} ω-dependent. For simplicity, this weaker version is considered.

Via Assumption 1.12 it is possible to prove the unique solvability of the AVP with random data (Problem 1.11) sample-wise using the Lax-Milgram lemma (Lemma 1.4) in the standard way; see [2]. Owing to Assumption 1.12, the coercivity and continuity constants in the Lax-Milgram lemma do not depend on ω.

1.3.2 FEM for the elliptic BVP with random data

In order to solve the AVP with random data (Problem 1.11) we use the finite element method. The FEM is used to solve the problem with a fixed value of the parameter $\omega \in \Omega$ after the parameter has been sampled. Thus there is, in principle, no difference from the deterministic case in the way how the FEM is used. Let us proceed to the formulation of the discrete problem.

Problem 1.13 (Discrete AVP with random data)**.** Let $V_{h,0} \subset V$ be the FE space (1.3). If $u_h(\cdot,\omega) \in V_{h,0}$ is such that

$$\tilde{a}(u_h, v_h, \omega) = l(v_h, \omega) \quad \forall v \in V_{h,0},$$

for a.e. $\omega \in \Omega$, then the function u_h is called a solution of the *discrete AVP with random data*. The functions $\tilde{a}$, l, and the parameter $\omega \in \Omega$ are taken from the AVP with random data (Problem 1.11).

The existence and uniqueness of the solution of the discrete AVP with random data follow from the inclusion $V_{h,0} \subset V$ and the fact that the non-discrete Problem 1.11 is uniquely solvable. Analogously to the non-discrete case, the coercivity and continuity constants are ω-independent due to Assumption 1.12.

1.3.3 Convergence results of the FEM with random data

In this section we will use the convergence results for the deterministic FEM from Section 1.2.4 to derive convergence results for the problem with random data, which will be useful in the subsequent chapters.

The following assumption is a stochastic version of Assumption 1.6. The assumption can be somewhat weakened by making the Lipschitz constant L ω-dependent.

Assumption 1.14. Consider the AVP (Problem 1.11). The domain D is assumed to be convex and the random field a is assumed to be uniformly Lipschitz continuous in the following sense. There exists $L > 0$ such that for a.e. $x_1, x_2 \in D$ and for a.e. $\omega \in \Omega$ it holds that

$$|a(x_1,\omega) - a(x_2,\omega)| \leq L\|x_1 - x_2\|.$$

This assumption allows us now to prove an estimate in the L^2 norm for the FEM with random data. Let us now formulate the probabilistic counterpart of Lemma 1.7 regarding the convergence of the FEM in the L^2 norm. The proof of this result is similar to the proof of Lemma 1.7. We present it here to stress how we use Assumption 1.14 and the fact that the constant L in this assumption is ω-independent.

Lemma 1.15 (Convergence of the FEM with random data in the L^2 norm)**.** *Let $\{\tau_h\}$ be a system of triangulations of the domain D. Let $u_h : D \times \Omega \to \mathbb{R}$ be such that for a.e. $\omega \in \Omega$ the function $u_h(\cdot,\omega) \in V_{h,0}$ is the solution of the discrete AVP with random data (Problem 1.13) corresponding to the triangulation τ_h. Then for a.e. $\omega \in \Omega$ the following estimate holds:*

$$\|u(\cdot,\omega) - u_h(\cdot,\omega)\|_{L^2(D)} \leq Ch^2\|f(\cdot,\omega)\|_{L^2(D)},$$

where $C > 0$ is independent of h, f, and ω.

Proof. In a standard way, via the Aubin-Nitsche duality trick, we can get

$$\|u(\cdot,\omega) - u_h(\cdot,\omega)\|_{L^2(D)} \leq Ch^2|u(\cdot,\omega)|_{H^2(D)}. \tag{1.10}$$

Here C is independent of h and u and because of Assumption 1.12, C is also independent of ω. This is because (1.10) is derived using the a-priori estimate

from the Lax-Milgram lemma (Lemma 1.4) and both the continuity constant and the coercivity constant α are independent of ω due to Assumption 1.12. Assumption 1.14 ensures that the solution u is H^2-regular in the following sense. There exists $C > 0$ such that for a.e. $\omega \in \Omega$ and for every $f(\cdot, \omega) \in L^2(D)$ we have

$$|u(\cdot, \omega)|_{H^2(D)} \leq C \|f(\cdot, \omega)\|_{L^2(D)}. \tag{1.11}$$

The ω-independence of the generic constant C is guaranteed here by the ω-independent constant L in Assumption 1.14. The claim of the lemma is now a consequence of the inequalities (1.10) and (1.11). □

References for the inequalities used in the proof were given in the proof of Lemma 1.7. Note that this theorem is a special case of Theorem 2.1 from [26]. The assumptions of this theorem are satisfied due to Assumptions 1.12 and 1.14 and the fact that we consider the homogeneous boundary condition.

Theorem 1.16 (FEM error for a functional of the solution)**.** *Let $p \in [1, \infty)$. Let $\{\tau_h\}$ be a system of triangulations of the domain D. Let $u_h : D \times \Omega \to \mathbb{R}$ be the same as in Lemma 1.15 and let the function $\omega \mapsto \|f(\cdot, \omega)\|_{L^2(D)}$ belong to the space $L^p(\Omega)$. Further, let $G : H_0^1(D) \to \mathbb{R}$ be a functional ω-uniformly Lipschitz continuous with respect to the L^2 norm, i.e.,*

$$\begin{aligned} &\exists L > 0 \, \forall u_1, u_2;\, u_1(\cdot, \omega), u_2(\cdot, \omega) \in H_0^1(D) \text{ for a.e. } \omega \in \Omega : \\ &\quad \Big|G(u_1(\cdot, \omega)) - G(u_2(\cdot, \omega))\Big| \leq L \|u_1(\cdot, \omega) - u_2(\cdot, \omega)\|_{L^2(D)}. \end{aligned} \tag{1.12}$$

Under these assumptions, the following estimate holds:

$$\|G(u) - G(u_h)\|_{L^p(\Omega, \mathbb{R})} \leq C h^2,$$

where C is independent of h, u, and ω. Here $G(u) - G(u_h)$ denotes the function $\omega \mapsto \Big(G(u(\cdot, \omega)) - G(u_h(\cdot, \omega))\Big)$.

Proof. First, it makes sense for the functional G to be Lipschitz continuous with respect to the L^2 norm, since $H_0^1(D) \hookrightarrow L^2(D)$. For a.e. $\omega \in \Omega$ we have, using the Lipschitz continuity of G,

$$\Big|G(u(\cdot, \omega)) - G(u_h(\cdot, \omega))\Big| \leq L \|u(\cdot, \omega) - u_h(\cdot, \omega)\|_{L^2(D)},$$

where L does not depend on ω. We can now use Lemma 1.15 to obtain

$$\Big|G(u(\cdot, \omega)) - G(u_h(\cdot, \omega))\Big| \leq C h^2 \|f(\cdot, \omega)\|_{L^2(D)}, \tag{1.13}$$

where C is a generic constant independent on ω and f. If we take equation (1.13) to the power p and integrate over Ω, we get

$$\|G(u) - G(u_h)\|^p_{L^p(\Omega, \mathbb{R})} \leq C^p h^{2p} \int_\Omega \|f(\cdot, \omega)\|^p_{L^2(D)} \mathrm{d}\omega,$$

which yields the desired inequality. □

Let us note that if the bounds in Assumption 1.12 are weakened and depend on ω, the analysis becomes more complicated and additional smoothness assumptions on the input data have to be made; see [26] for a thorough analysis. We now give an example of a functional satisfying the Lipschitz condition from Lemma 1.16.

Example 1.17. Let us prove that the functional $G : H_0^1(D) \to \mathbb{R}$ defined by

$$G(v) \coloneqq \int_D v(x)\mathrm{dx}$$

is uniformly Lipschitz continuous with respect to the L^2 norm in the sense of (1.12).

Indeed, let us denote by $\lambda(D)$ the Lebesgue measure of D and define $M \coloneqq \sqrt{\lambda(D)}$. Let $u_1, u_2 : \Omega \times D \to \mathbb{R}$ be such that $u_1(\cdot,\omega), u_2(\cdot,\omega) \in H_0^1(D)$ for a.e. $\omega \in \Omega$. Then

$$\begin{aligned}\left|G(u_1(\cdot,\omega)) - G(u_2(\cdot,\omega))\right| &= \left|\int_D u_1(x,\omega) - u_2(x,\omega)\mathrm{dx}\right| \\ &\leq \int_D |u_1(x,\omega) - u_2(x,\omega)|\mathrm{dx} \\ &\leq \|u(\cdot,\omega) - u_2(\cdot,\omega)\|_{L^2(D)}\|1\|_{L^2(D)} \\ &= M\|u(\cdot,\omega) - u_2(\cdot,\omega)\|_{L^2(D)},\end{aligned}$$

where in the last inequality we used Hölder's inequality. The result we obtained also proves that the functional is well defined, since $H_0^1(D) \hookrightarrow L^2(D)$.

1.3.4 FEM with random data in finite precision arithmetic

The goal of this section is to obtain a result analogous to Lemma 1.8 in our probabilistic setup and then, as a consequence, a similar result for the functional $G(u)$. At first, we straightforwardly generalize the notions from Section 1.2.5.

Let $\widehat{u}_h$ be the approximation of the FE solution u_h to the discrete AVP with random data (Problem 1.13) such that

$$\widehat{u}_h(\cdot,\omega) = \sum_{j=1}^{n} \widehat{x}_j(\omega)\phi_j.$$

Similarly we define the residual r as $r(\omega) \coloneqq A(\omega)\widehat{x}(\omega) - b(\omega)$. Let us now state a lemma analogous to Lemma 1.8.

Lemma 1.18. *Let u_h be the solution of the discrete AVP with random data (Problem 1.13) and let $\widehat{u}_h$ be the approximation of u_h defined above. Then for a.e. $\omega \in \Omega$ it holds that*

$$\|u_h(\cdot,\omega) - \widehat{u}_h(\cdot,\omega)\|_{H_0^1(D)} \leq C\|f(\cdot,\omega)\|_{L^2(D)}\frac{\|r(\omega)\|_{\mathbb{R}^n}}{\|b(\omega)\|_{\mathbb{R}^n}},$$

where C is independent of h, u, and ω.

Proof. For a fixed $\omega \in \Omega$ we can proceed completely analogously to the proof of Lemma 1.8. The only difference is that in this case the RHS f is ω-dependent and thus it cannot be included in the constant C.

The fact that C is independent of h follows similarly as in Theorem 1.8, since we assume the family of triangulations $\{\tau_h\}$ to be shape-regular. Similarly in this case the bilinear form $\tilde{a}_h$ is uniformly bounded with respect to h and ω due to the fact that $\tilde{a}_h = \tilde{a}$ and Assumption 1.12.

To verify that C is also independent of ω, we write down the constant explicitly (see Proposition 9.19 in [10]). It holds that

$$C = \frac{\kappa(\mathcal{M}_t)^{1/2}}{c_{tP}\alpha}. \tag{1.14}$$

The condition number $\kappa(\mathcal{M}_t)$ is independent of ω from the definition of $\mathcal{M}_t$, see the first paragraph of Section 9.1.3 in [10]. Further, α is the coercivity constant and it is independent of ω from Assumption 1.12 and c_{tP} is the constant from the embedding $H_0^1(D) \hookrightarrow L^2(D)$. This completes the proof. □

This theorem allows us to control the error $\|u_h(\cdot,\omega) - \widehat{u}_h(\cdot,\omega)\|_{H_0^1(D)}$ if we control the relative residual $\|r(\omega)\|_{\mathbb{R}^n}/\|b(\omega)\|_{\mathbb{R}^n}$ of the resulting linear system. We proceed to the estimate of the error in the case of the functional $G(u)$.

Lemma 1.19. *Let u_h, $\widehat{u}_h$, and r be as in Lemma 1.18. Further, let $G : H_0^1(D) \to \mathbb{R}$ be a functional ω-uniformly Lipschitz continuous with respect to the L^2 norm (see Theorem 1.16). Then for a.e. $\omega \in \Omega$,*

$$\left|G(u_h(\cdot,\omega)) - G(\widehat{u}_h(\cdot,\omega))\right| \leq C\|f(\cdot,\omega)\|_{L^2(D)} \frac{\|r(\omega)\|_{\mathbb{R}^n}}{\|b(\omega)\|_{\mathbb{R}^n}},$$

where the constant C is independent of ω, u, and h.

Proof. We have

$$\begin{aligned}\left|G(u_h(\cdot,\omega)) - G(\widehat{u}_h(\cdot,\omega))\right| &\leq L\|u(\cdot,\omega) - \widehat{u}_h(\cdot,\omega)\|_{L^2(D)} \\ &\leq C\|u(\cdot,\omega) - \widehat{u}_h(\cdot,\omega)\|_{H_0^1(D)} \\ &\leq C\|f(\cdot,\omega)\|_{L^2(D)} \frac{\|r(\omega)\|_{\mathbb{R}^n}}{\|b(\omega)\|_{\mathbb{R}^n}},\end{aligned}$$

where we used the Lipschitz continuity of G, the Poincaré-Friedrichs inequality, and Lemma 1.18, respectively. The generic constant C is independent of ω, u, and h due to Lemma 1.18 and the proof is complete. □

2. Monte Carlo methods

Our ultimate goal is to approximate a quantity of interest given as a function of the solution of a PDE with random input data. The quantity of interest will in our case be given by the expected value of a functional of the solution. Chapter 1 was concerned with the discretisation of the underlying PDE in the spatial variable. After the PDE has been discretised, we have to face the challenge of approximating the integral in the expected value. A more general question is how uncertainties in the input data (i.e., in the coefficients of the PDE) influence the results. This is the concern of the field called uncertainty quantification (UQ). Uncertainty quantification uses various types of methods to determine how the results of the models are affected by uncertain input data, one of them being the Monte Carlo (MC) class of methods. This broad class of methods will be our focus now. This chapter provides an explanation of the basic principles and convergence results of two Monte Carlo methods, namely, the standard MC method and the MLMC method, focusing mostly on problems coming from PDEs. Note that in this chapter, we do not take into account any effect of rounding errors and finite precision computations. The impact of the use of finite precision arithmetic and how to exploit it will be discussed in Chapter 4. This chapter is based on [14] and [6].

We begin this chapter by introducing our model problem, i.e., we formulate precisely the problem of approximating the quantity of interest. The Monte Carlo methods described below will be, for simplicity, demonstrated and explained using this problem. Note that the MC methods can be used to tackle various types of problems; see [14] for an overview. We restrict ourselves to this model problem to show the basic principles.

Problem 2.1. Consider the elliptic boundary value problem with random data (Problem 1.10). Let $G : H_0^1(D) \to \mathbb{R}$ be a functional and let u be the weak solution of the BVP (1.9). Estimate the quantity of interest (QOI) defined as the expected value of the random variable $Q : \Omega \to \mathbb{R}$ given by $\omega \mapsto G(u(\cdot, \omega))$. The functional G is assumed to be uniformly Lipschitz continuous in the sense of (1.12).

The expected value $\mathbb{E}[Q]$ from the definition above will also be denoted by $\mathbb{E}[G(u)]$. Remark that there are other possibilities for what the quantity of interest can be, e.g., the mean $\mathbb{E}[u(x, \cdot)]$ or the variance $\mathrm{var}[u(x, \cdot)]$. The class of MC methods can be applied to tackle these problems as well. We will for simplicity keep our focus on Problem 2.1.

2.1 Monte Carlo method with analytic solution

This section describes in more detail the basic Monte Carlo method. The idea behind the Monte Carlo estimator is simple — the QOI is approximated by the sample mean and we expect the sample mean to converge to the expected value. For simplicity, we begin with the case where the analytic solution of the underlying PDE is given. We proceed directly to the definition of the MC method for Problem 2.1.

Definition 2.2 (Monte Carlo estimator with analytic solution)**.** *Let $u : D \times \Omega \to \mathbb{R}$ be the weak solution of the BVP* (1.9)*. Let $\mathbb{E}[Q]$ be the expected value (quantity of interest) defined in Problem 2.1. Define the Monte Carlo estimator for $E[Q]$ by*

$$\widehat{Q}_N \coloneqq \frac{1}{N} \sum_{k=1}^{N} Q^{(k)},$$

where $Q^{(k)}$ are independent and identically distributed (i.i.d.) random variables following the same distribution as Q.

In this first case, we are able to evaluate $Q^{(k)}(\omega)$ exactly for $\omega \in \Omega$, since we are given the analytic solution u of the PDE and we assume that the functional G can be evaluated exactly. In what follows, we will use the expression "random samples of Q" instead of "i.i.d. random variables following the same distribution as Q". In order to quantify the error of the estimator we will use the mean square error (MSE), this will be the case for all of the following estimators. Since $\widehat{Q}_N$ is an unbiased estimator for $\mathbb{E}[Q]$, the mean square error of $\widehat{Q}_N$ can be expressed via the variance of Q, as stated in the following lemma. The proofs of these auxiliary results for the MC method will be given here, since they illustrate the basic principles which are then applied in more complex cases.

Lemma 2.3 ("Bias-variance decomposition" of the MSE of the Basic MC estimator)**.** *The mean square error of the MC estimator $\widehat{Q}_N$ from Definition 2.2 can be expanded as*

$$\mathbb{E}\Big[(\mathbb{E}[Q] - \widehat{Q}_N)^2\Big] = \frac{var[Q]}{N}.$$

Proof. From the basic properties of the mean we can write

$$\begin{aligned}\mathbb{E}\Big[(\mathbb{E}[Q] - \widehat{Q}_N)^2\Big] &= \mathbb{E}\Big[(\mathbb{E}[Q])^2 - 2\mathbb{E}[Q]\widehat{Q}_N + (\widehat{Q}_N)^2\Big] \\ &= (\mathbb{E}[Q])^2 - 2\mathbb{E}[Q]\mathbb{E}[\widehat{Q}_N] + \mathbb{E}[(\widehat{Q}_N)^2].\end{aligned}$$

Since the $\widehat{Q}_N$ is an unbiased estimator for $\mathbb{E}[Q]$, the last expression can be rewritten as

$$(\mathbb{E}[Q])^2 - 2\mathbb{E}[Q]\mathbb{E}[\widehat{Q}_N] + \mathbb{E}[(\widehat{Q}_N)^2] = \mathbb{E}[(\widehat{Q}_N)^2] - (\mathbb{E}[\widehat{Q}_N])^2 = \mathrm{var}[\widehat{Q}_N].$$

Finally, since $Q^{(k)}$ are i.i.d., we conclude using the basic properties of the variance that $\mathbb{E}\Big[(\mathbb{E}[Q] - \widehat{Q}_N)^2\Big] = \mathrm{var}[\widehat{Q}_N] = \frac{\mathrm{var}[Q]}{N}$. □

Even though the lemma is called "Bias-variance decomposition", there is no term corresponding to the bias in the expression of the MSE. That is because in this case we are assumed to have the analytic solution of the PDE which allows the estimator to be unbiased. This is, however, unrealistic in practical problems. When we do not have the analytic solution and use the FEM to solve the PDE, the decomposition will contain a term corresponding to the bias. Such a case will be discussed in the following section.

2.2 Monte Carlo method

In this section we no longer assume that we know the analytic solution of the PDE, and instead we use the FEM to solve the PDE to obtain the discrete solution. This will make our MC estimator biased. The discrete solution does not have to be obtained by the conforming FE method; in principle it can be obtained by any convenient discretisation method. Even though we will, for simplicity, use the FEM defined in Chapter 1, sometimes much more general results can be obtained with no additional effort. In such situations we present the more general results. The Monte Carlo estimator which uses the discrete solution is defined as follows:

Definition 2.4 (Monte Carlo estimator)**.** *Let $u_h : D \times \Omega \to \mathbb{R}$ be the solution of the discrete AVP with random data (Problem 1.13) or any other discrete solution of the AVP (Problem 1.11). Let $\mathbb{E}[Q]$ be the quantity of interest defined in Problem 2.1. Let Q_h be a random variable defined as $Q_h : \Omega \to \mathbb{R},\ \omega \mapsto G(u_h(\cdot, \omega))$. Define the Monte Carlo estimator for $E[Q]$ by*

$$\widehat{Q}_{h,N} \coloneqq \frac{1}{N} \sum_{k=1}^{N} Q_h^{(k)},$$

where $Q_h^{(k)}$ are random samples from Q_h.

We can immediately see that the MC estimator is an unbiased estimator for $\mathbb{E}[Q_h]$. However, it is not an unbiased estimator for $\mathbb{E}[Q]$. In this case, the MSE can be expanded as follows.

Lemma 2.5 (Bias-variance decomposition of the MSE of the MC estimator)**.** *The mean square error of the MC estimator $\widehat{Q}_{h,N}$ from Definition 2.4 can be expanded as*

$$\mathbb{E}\Big[(\mathbb{E}[Q] - \widehat{Q}_{h,N})^2\Big] = (\mathbb{E}[Q - Q_h])^2 + \frac{var[Q_h]}{N}.$$

Proof. We rewrite the left-hand side to obtain

$$\mathbb{E}\Big[(\mathbb{E}[Q] - \widehat{Q}_{h,N})^2\Big] = \mathbb{E}\Big[\Big((\mathbb{E}[Q] - \mathbb{E}[Q_h]) + (\mathbb{E}[Q_h] - \widehat{Q}_{h,N})\Big)^2\Big].$$

The next steps are similar to the proof of Lemma 2.3:

$$\begin{aligned}
&\mathbb{E}\Big[\Big((\mathbb{E}[Q] - \mathbb{E}[Q_h]) + (\mathbb{E}[Q_h] - \widehat{Q}_{h,N})\Big)^2\Big] \\
&= (\mathbb{E}[Q - Q_h])^2 + 2\mathbb{E}[Q - Q_h]\mathbb{E}\Big[\mathbb{E}[Q_h] - \widehat{Q}_{h,N}\Big] + \mathbb{E}\Big[(\mathbb{E}[Q_h] - \widehat{Q}_{h,N})^2\Big].
\end{aligned}$$

Since $\widehat{Q}_{h,N}$ is an unbiased estimator for $\mathbb{E}[Q_h]$, the last expression can be rewritten as follows:

$$\begin{aligned}
&(\mathbb{E}[Q - Q_h])^2 + 2\mathbb{E}[Q - Q_h]\mathbb{E}\Big[\mathbb{E}[Q_h] - \widehat{Q}_{h,N}\Big] + \mathbb{E}\Big[(\mathbb{E}[Q_h] - \widehat{Q}_{h,N})^2\Big] \\
&= (\mathbb{E}[Q - Q_h])^2 + \mathbb{E}\Big[(\mathbb{E}[\widehat{Q}_{h,N}] - \widehat{Q}_{h,N})^2\Big].
\end{aligned}$$

The above expression is now equal to $\mathrm{var}[\widehat{Q}_{h,N}]$. Using the fact that the $Q_h^{(k)}$ are i.i.d., we can conclude that $\mathrm{var}[\widehat{Q}_{h,N}] = \frac{\mathrm{var}[Q_h]}{N}$, which completes the proof. □

Note that if in Lemma 2.5 we take the analytic solution u instead of the discrete solution u_h, the random variable Q_h reduces to Q and we are back in the setting of the Bias-variance decomposition of the MC estimator (Lemma 2.3).

In practical applications, we want to ensure that the MSE is under a given tolerance. The question is what the total cost of computing the MC estimate will then be — we want to express the cost in terms of the tolerance. The answer to this question is provided by the following theorem. The theorem is abstract in the sense that the discrete solution does not have to be the piecewise linear FE solution as defined in Chapter 1; it just has to satisfy some given assumptions. The version of the theorem specifically for the FE solution is given below. Note that throughout the thesis, the cost will be measured in terms of floating-point operations (FLOPs).

Theorem 2.6 (Abstract complexity theorem for the MC method)**.** *Assume that there exist $\alpha, \gamma > 0$ such that*

$$|\mathbb{E}[Q_h - Q]| = O(h^\alpha) \quad \text{for } h \to 0, \tag{2.1}$$

$$\mathit{Cost}(Q_h^{(k)}) = O(h^{-\gamma}) \quad \text{for } h \to 0, \tag{2.2}$$

where Q_h is from Definition 2.4 and $\mathit{Cost}(Q_h^{(k)})$ denotes the cost per one sample from the random variable Q_h in terms of FLOPs. Assume further there exists σ^2 such that for all $h > 0$ sufficiently small it holds $var[Q_h] \leq \sigma^2$. Then for any $\epsilon > 0$ sufficiently small there exist $N \in \mathbb{N}$ and $h > 0$ such that the MSE is bounded by ϵ^2, i.e.,

$$\|\mathbb{E}[Q] - \widehat{Q}_{h,N}\|^2_{L^2(\Omega,\mathbb{R})} = \mathbb{E}\Big[(\mathbb{E}[Q] - \widehat{Q}_{h,N})^2\Big] \leq \epsilon^2$$

and the total computational cost of the MC estimate for $\mathbb{E}[Q]$ satisfies

$$\mathit{Cost}(\widehat{Q}_{h,N}) = O(\epsilon^{-2-\gamma/\alpha}).$$

Proof. The basic idea of the proof is to balance the two components in the bias-variance decomposition (Lemma 2.5). From (2.1) it follows that

$$\exists C_1 > 0 \, \forall \tilde{h} \in (0, h_1) : |\mathbb{E}[Q_h - Q]| \leq C_1 \tilde{h}^\alpha.$$

Let $\epsilon > 0$ be given. If ϵ is sufficiently small then we have

$$\exists h \in (0, h_1) : C_1 h^\alpha = \frac{\epsilon}{\sqrt{2}}. \tag{2.3}$$

Now, let N be defined by

$$N \coloneqq \Big\lceil 2 \operatorname{var}[Q_h] \epsilon^{-2} \Big\rceil. \tag{2.4}$$

Then it holds that

$$\frac{\operatorname{var}[Q_h]}{N} \leq \frac{\epsilon^2}{2}. \tag{2.5}$$

Using the bias-variance decomposition (Lemma 2.5), we obtain from (2.3) and (2.5) a bound for the MSE:

$$\mathbb{E}\Big[(\mathbb{E}[Q] - \widehat{Q}_{h,N})^2\Big] = (\mathbb{E}[Q - Q_h])^2 + \frac{\operatorname{var}[Q_h]}{N} \leq \frac{\epsilon^2}{2} + \frac{\epsilon^2}{2} \leq \epsilon^2.$$

Let us now determine the total cost. From (2.4) it follows that for ϵ sufficiently small we have

$$N \leq 4 \operatorname{var}[Q_h]\epsilon^{-2} \leq 4\sigma^2\epsilon^{-2} = O(\epsilon^{-2}). \tag{2.6}$$

Equation (2.3) yields $h = C\epsilon^{1/\alpha}$ for a generic constant C. From the assumption (2.2) it follows (if ϵ and consequently h are sufficiently small) that $\exists C_2 > 0$: $\operatorname{Cost}(Q_h^{(k)}) \leq C_2 h^{-\gamma}$. Using the fact that $h = C\epsilon^{1/\alpha}$, we obtain

$$\operatorname{Cost}(Q_h^{(k)}) = O(\epsilon^{-\gamma/\alpha}). \tag{2.7}$$

The overall cost can be expressed as $\operatorname{Cost}(\widehat{Q}_{h,N}) = N \cdot \operatorname{Cost}(Q_h^{(k)})$ which, along with (2.6) and (2.7), gives us the desired result. □

To formulate the MC complexity theorem specifically for the finite element solution, the computational cost of solving the FE system (1.6) has to be discussed so that we are able to determine the parameter γ from the previous lemma. This cost depends on the numerical solver we use to approximate the solution. In the best case the system can be solved in linear complexity, which can be achieved when an optimal multigrid solver is used. Throughout this chapter we assume that the cost of solving the FE system is $O(n)$ FLOPs, where n is the dimension of the matrix. This assumption leads to the constant $\gamma = 2$ in Theorem 2.6, since $n \approx h^{-2}$, where h is the discretisation parameter. Naturally, the multigrid solver is not the only possibility for solving the FE system. The FE system (1.6) can be solved, e.g., by a sparse direct solver based on Cholesky factorization. In this case the theoretical complexity of solving the system is $O(n^{3/2})$ operations assuming $D \subset \mathbb{R}^2$; see Section 3.2.1.

Theorem 2.7 (Monte Carlo FEM complexity)**.** *Let f be sufficiently smooth so that Theorem 1.16 holds and assume that the cost per sample increases as*

$$\mathit{Cost}(Q_h^{(k)}) = O(h^{-2}).$$

Then for any $\epsilon > 0$ there exist $h > 0$ and $N \in \mathbb{N}$ such that

$$\mathbb{E}\Big[(\mathbb{E}[Q] - \widehat{Q}_{h,N})^2\Big] \leq \epsilon^2$$

and

$$\mathit{Cost}(\widehat{Q}_{h,N}) = O(\epsilon^{-3}),$$

where $\widehat{Q}_{h,N}$ is the MC estimator for $\mathbb{E}[Q]$ in the sense of Definition 2.4 corresponding to the solution u_h of the discrete AVP with random data (Problem 1.13).

Proof. We will show that we can apply Theorem 2.6 with $\alpha = 2$ and $\gamma = 2$. Let us at first prove the boundedness of the quantity $\operatorname{var}[Q_h]$:

$$\begin{aligned}
\operatorname{var}[Q_h] &= \mathbb{E}[Q_h^2] - (\mathbb{E}[Q_h])^2 \\
&\leq \mathbb{E}[(Q_h - Q + Q)^2] \\
&= \mathbb{E}[(Q_h - Q)^2] + 2\mathbb{E}[(Q_h - Q)Q] + \mathbb{E}[Q^2] \\
&\leq 2\mathbb{E}[(Q_h - Q)^2] + 2\mathbb{E}[Q^2] \\
&\leq Ch^4 + 2\mathbb{E}[Q^2] \\
&\leq Ch_{\text{bnd}}^4 + 2\mathbb{E}[Q^2],
\end{aligned}$$

where we have used Theorem 1.16. The fact that $\alpha = 2$ can be deduced from Theorem 1.16:

$$|\mathbb{E}[Q_h - Q]| \leq \mathbb{E}[|G(u_h) - G(u)|] = \|G(u) - G(u_h)\|_{L^1(\Omega)} = O(h^2),$$

where we used Jensen's inequality, the definition of the mean, and Theorem 1.16, respectively. Thus Assumption (2.1) holds with $\alpha = 2$.

Finally, Assumption (2.2) holds with $\gamma = 2$ as assumed, which concludes the proof. □

If a different solver than multigrid is used to solve the FE system, we obtain in general a different value of the parameter γ and the result of the complexity theorem (Theorem 2.7) has to be adjusted accordingly.

2.3 Multilevel Monte Carlo Method

Another of the MC-type methods is the so-called Multilevel Monte Carlo (MLMC) method, which was first introduced in the context of stochastic differential equations in 2008 by Mike Giles in the pioneering work [13]. It was soon applied in uncertainty quantification for PDE-related problems. One of the first works in this context was [6], which we follow here. MLMC is a refinement of the traditional Monte Carlo method that aims to reduce the computational cost of the estimator by sampling on a hierarchy of discretisation levels. The underlying idea is the following: Let us consider $L + 1$ discretisation levels labeled $0, \ldots, L$ and denote by $\mathbb{E}[Q_l]$ the approximation of the quantity of interest $\mathbb{E}[Q]$ on level l. The meshes corresponding to the individual discretisation levels are assumed to be refined with increasing l. Then we can write, using the telescopic sum,

$$\mathbb{E}[Q_L] = \mathbb{E}[Q_0] + \sum_{l=1}^{L} \mathbb{E}[Q_l - Q_{l-1}].$$

Each term in the sum can now be estimated by a simple Monte Carlo estimator. Since the variance of the random variables $\mathbb{E}[Q_l - Q_{l-1}]$ decays quickly for increasing l, it is sufficient to take only a small number of samples on the finer levels of the discretisation and still, due to the telescopic sum, obtain the bias error corresponding to the finest level L. The precise formulation is given below. In our work, we will use MLMC extensively in combination with mixed-precision computations.

At first, we present the definition of the MLMC estimator and then the corresponding bias-variance decomposition theorem. With the bias-variance decomposition in hand, it will be then easier to explain the underlying principles behind the MLMC method and how the computational cost is reduced compared to the MC method.

2.3.1 Definition and complexity theorem

Definition 2.8 (Multilevel Monte Carlo estimator)**.** *Let $\mathbb{E}[Q]$ be the quantity of interest defined in Problem 2.1. Let $h_0 \geq \ldots \geq h_L > 0$ be the discretisation*

parameters, u_{h_l} the corresponding solutions of the discrete random AVP (Problem 1.13) and Q_{h_l}, $l \in \{0, \dots, L\}$, the corresponding random variables defined analogously as in Definition 2.4. Define the following auxiliary MC estimators:

$$\widehat{Y}_0 = \widehat{Y}_{h_0,N_0} \coloneqq \frac{1}{N_0} \sum_{k=1}^{N_l} Q_{h_0}^{(k)},$$
$$\widehat{Y}_l = \widehat{Y}_{h_l,N_l} \coloneqq \frac{1}{N_l} \sum_{k=1}^{N_l} \Big(Q_{h_l}^{(k)} - Q_{h_{l-1}}^{(k)} \Big), \quad l = 1 \dots, L.$$

Then, the estimator

$$\widehat{Q}_{L,\{N_l\}}^{ML} \coloneqq \sum_{l=0}^{L} \widehat{Y}_l$$

will be referred to as the MLMC estimator for $E[Q]$.

This means that the $\widehat{Y}_l$ are in fact MC estimators for the expected values of random variables Y_l defined by $Y_0 \coloneqq Q_{h_0}$ and $Y_l \coloneqq (Q_{h_l} - Q_{h_{l-1}})$ for $l \geq 1$. Important is that the estimators $\widehat{Y}_l$ are, in this work, assumed to be independent. Note that when computing the estimate of $\mathbb{E}[Y_l] = E[Q_{h_l} - Q_{h_{l-1}}]$ using $\widehat{Y}_l$, the difference is computed for each k using a single sample $\omega^{(k)}$. We can now proceed to the bias-variance decomposition of the MLMC estimator.

Lemma 2.9 (Bias-variance decomposition for the MLMC estimator)**.** *The mean square error of the MLMC estimator $\widehat{Q}_{L,\{N_l\}}^{ML}$ from Definition 2.8 can be expanded as*

$$\mathbb{E}\Big[(\mathbb{E}[Q] - \widehat{Q}_{L,\{N_l\}}^{ML})^2 \Big] = (\mathbb{E}[Q - Q_{h_L}])^2 + \sum_{l=0}^{L} \frac{var[Y_l]}{N_l}.$$

Proof. The mean of the MLMC estimator can be rewritten as

$$\mathbb{E}[\widehat{Q}_{L,\{N_l\}}^{\mathrm{ML}}] = \sum_{l=0}^{L} \mathbb{E}[\widehat{Y}_l] = \mathbb{E}[Q_{h_0}] + \sum_{l=1}^{L} \mathbb{E}[Q_{h_l} - Q_{h_{l-1}}] = E[Q_{h_L}],$$

since $\widehat{Y}_l$ are unbiased estimators of Y_l and the sum is telescopic. We can now exploit the fact that $\widehat{Y}_l$ are independent to get

$$\mathrm{var}[\widehat{Q}_{L,\{N_l\}}^{\mathrm{ML}}] = \sum_{l=0}^{L} \mathrm{var}[\widehat{Y}_l].$$

Analogously to the proof of Lemma 2.5 we obtain

$$\sum_{l=0}^{L} \mathrm{var}[\widehat{Y}_l] = \sum_{l=0}^{L} \frac{\mathrm{var}[Y_l]}{N_l},$$

and the proof is finished. □

We now give an intuitive explanation of why the computational cost is reduced using the MLMC method compared to the MC method while preserving the error tolerance. Let us have a look at the bias-variance decompositions for the MC method and the MLMC method given by Lemma 2.5 and Lemma 2.9, respectively. In order to get the same bias error in the MC method as in the MLMC method,

we have to take $h \coloneqq h_L$. In order to balance the bias and the variance terms in the bias variance decomposition of the MC method, we have to take a large number of samples N_L to make the term $\frac{\mathrm{var}[Q_{h_L}]}{N_L}$ in Lemma 2.5 small. This is not the case in the MLMC method, because the variance $\mathrm{var}[Y_l]$ decays rapidly with increasing l and we can thus take only a small number of samples N_L on the finest level of discretisation where the computation is most expensive. The precise meaning of this intuitive explanation is given in the following theorem.

Theorem 2.10 (Abstract complexity theorem for the MLMC method)**.** *Let $m \in \mathbb{N}$, $m > 1$, and let $h_0, h_1, \dots$ be discretisation parameters satisfying $h_0 > 0$ and $h_l = \frac{1}{m} h_{l-1}$. Assume that there exist $\alpha, \beta, \gamma > 0$ such that $\alpha \geq \frac{1}{2}\min\{\beta, \gamma\}$ and*

$$|\mathbb{E}[Q_{h_l} - Q]| = O(h_l^{\alpha}), \tag{2.8}$$

$$var[Y_l] = O(h_l^{\beta}), \tag{2.9}$$

$$C_l = O(h_l^{-\gamma}), \tag{2.10}$$

where $C_l \coloneqq Cost(Y_l^{(k)})$ and the notation is the same as in Definition 2.8. Then for any $0 < \epsilon < \exp^{-1}$ there exist $L \in \mathbb{N}$ and $\{N_l\}_{l=0}^{L}$ such that

$$\mathbb{E}\Big[(\mathbb{E}[Q] - \widehat{Q}^{ML}_{L,\{N_l\}})^2\Big] < \epsilon^2,$$

and the total computational cost of a MLMC estimate satisfies

$$Cost(\widehat{Q}^{ML}_{L,\{N_l\}}) = \begin{cases} O(\epsilon^{-2}), & \text{if } \beta > \gamma, \\ O(\epsilon^{-2}|\log\epsilon|^2), & \text{if } \beta = \gamma, \\ O(\epsilon^{-2-(\gamma-\beta)/\alpha}), & \text{if } \beta < \gamma. \end{cases}$$

Proof. The proof is adapted from [6]. Let $\epsilon > 0$. Let us denote by c_1, c_2, c_3 the constants from (2.8), (2.9), and (2.10), respectively. We can without loss of generality (WLOG) assume that $h_0 = 1$, otherwise we just scale the constants c_1, c_2, c_3. Under this assumption we can write $h_l = m^{-l}$. In order to bound the mean squared error by ϵ^2, we balance the terms in the bias-variance decomposition (Lemma 2.9). Both the bias term and the variance term will be bounded by $\frac{\epsilon^2}{2}$.

Let $L \in \mathbb{N}$ be defined as

$$L \coloneqq \lceil \alpha^{-1} \log_m(\sqrt{2} c_1 \epsilon^{-1}) \rceil.$$

Then it holds that

$$m^{-\alpha}\frac{\epsilon}{\sqrt{2}} < c_1 m^{-L\alpha} \leq \frac{\epsilon}{\sqrt{2}}, \tag{2.11}$$

because the right-hand side inequality is equivalent to $L \geq \alpha^{-1}\log_m(\sqrt{2}c_1\epsilon^{-1})$ and the left-hand side inequality in (2.11) is equivalent to $L < \alpha^{-1}\log_m(\sqrt{2}c_1\epsilon^{-1}) + 1$. Using the second inequality in (2.11) we get

$$|\mathbb{E}[Q_{h_L} - Q]| \leq c_1 h_L^{\alpha} = c_1 m^{-L\alpha} \leq \frac{\epsilon}{\sqrt{2}}, \tag{2.12}$$

and the bias error is bounded.

We will now derive an auxiliary inequality which will be used later. The formula for the sum of the geometric sequence and a straightforward computation yield

$$\sum_{l=0}^{L} m^{\gamma l} = \frac{(m^\gamma)^{L+1}-1}{m^\gamma - 1} = \frac{m^{\gamma L} - m^{-\gamma}}{1-m^{-\gamma}} < \frac{m^{\gamma L}}{1-m^{-\gamma}}. \tag{2.13}$$

Using the first inequality in (2.11) we obtain

$$\sum_{l=0}^{L} m^{\gamma l} < \frac{m^{\gamma L}}{1-m^{-\gamma}} < \frac{m^\gamma (\sqrt{2}c_1)^{\frac{\gamma}{\alpha}}}{1-m^{-\gamma}} \epsilon^{-\frac{\gamma}{\alpha}}. \tag{2.14}$$

Let us distinguish three cases:

1. First we consider $\beta = \gamma$. In this case, let us define N_l by

$$N_l \coloneqq \Big\lceil 2\epsilon^{-2}(L+1)c_2 m^{-\beta l} \Big\rceil.$$

It follows that

$$\operatorname{var}\Big[\widehat{Q}^{\mathrm{ML}}_{L,\{N_l\}}\Big] = \sum_{l=0}^{L} \operatorname{var}[\widehat{Y}_l] \le \sum_{l=0}^{L} \frac{c_2 m^{-\beta l}}{N_l} \le \frac{\epsilon^2}{2},$$

where we have used the definition of $\widehat{Q}^{\mathrm{ML}}_{L,\{N_l\}}$ and the fact that $\widehat{Y}_l$ are independent, the assumption (2.9), and the definition of N_l, respectively. It remains to determine the overall cost. We have

$$\begin{aligned}
\operatorname{Cost}(\widehat{Q}^{\mathrm{ML}}_{L,\{N_l\}}) &\le c_3 \sum_{l=0}^{L} N_l m^{\gamma l} \\
&\le c_3 \sum_{l=0}^{L} \Big(2\epsilon^{-2}(L+1)c_2 m^{-\beta l} + 1\Big) m^{\gamma l} \\
&\le c_3 \Big(2\epsilon^{-2}(L+1)^2 c_2 + \sum_{l=0}^{L} m^{\gamma l}\Big) \\
&\le C\Big(\epsilon^{-2}(\log\epsilon)^2 + \epsilon^{-\frac{\gamma}{\alpha}}\Big),
\end{aligned}$$

where $C > 0$ is a generic constant. Here we used the assumption (2.10), the definition of N_l, the fact that $\beta = \gamma$, the definition of L, and the inequality (2.14), respectively. Since $\epsilon < e^{-1}$, we have $1 = \Big(\log(e^{-1})\Big)^2 < \Big(\log(\epsilon)\Big)^2$. Together with $\alpha \ge \frac{\gamma}{2}$ we obtain

$$\epsilon^{-\frac{\gamma}{\alpha}} \le \epsilon^{-2} \le \epsilon^{-2}(\log\epsilon)^2$$

and in total,

$$\operatorname{Cost}(\widehat{Q}^{\mathrm{ML}}_{L,\{N_l\}}) \le C\Big(\epsilon^{-2}(\log\epsilon)^2 + \epsilon^{-\frac{\gamma}{\alpha}}\Big) = O(\epsilon^{-2}|\log\epsilon|^2).$$

2. Assume now $\beta < \gamma$. Since the idea of the proof remains the same, we will therefore proceed more briefly. Let

$$N_l \coloneqq \left\lceil 2\epsilon^{-2} c_2 m^{(\gamma-\beta)\frac{L}{2}} \frac{m^{-(\beta+\gamma)\frac{l}{2}}}{1-m^{-(\gamma-\beta)/2}} \right\rceil.$$

Analogously to the previous case we can estimate

$$\sum_{l=0}^{L} \mathrm{var}[\widehat{Y}_l] \leq \frac{\epsilon^2}{2} m^{-(\gamma-\beta)\frac{L}{2}} \Big(1 - m^{-(\gamma-\beta)/2}\Big) \sum_{l=0}^{L} m^{(\gamma-\beta)\frac{l}{2}} \leq \frac{\epsilon^2}{2},$$

where we used the definition of N_l and the inequality (2.13). The cost can be estimated by

$$\mathrm{Cost}(\widehat{Q}^{\mathrm{ML}}_{L,\{N_l\}}) < c_3 \left(2\epsilon^{-2} c_2 \frac{m^{(\gamma-\beta)\frac{L}{2}}}{1 - m^{-(\gamma-\beta)/2}} \sum_{l=0}^{L} m^{(\gamma-\beta)\frac{l}{2}} + \sum_{l=0}^{L} m^{\gamma l} \right). \quad (2.15)$$

The left-hand side inequality in (2.11) yields

$$m^{(\gamma-\beta)L} \leq \Big(\sqrt{2} c_1\Big)^{\frac{\gamma-\beta}{\alpha}} m^{\gamma-\beta} \epsilon^{-\frac{\gamma-\beta}{\alpha}}.$$

Also $\epsilon < e^{-1} < 1$ and $\alpha \geq \frac{1}{2}\beta$ and therefore $\epsilon^{-\frac{\gamma}{\beta}} \leq \epsilon^{-2-\frac{\gamma-\beta}{\alpha}}$. Using this and the (2.14), the inequality (2.15) yields

$$\mathrm{Cost}(\widehat{Q}^{\mathrm{ML}}_{L,\{N_l\}}) = O(\epsilon^{-2-\frac{\gamma-\beta}{2}}).$$

3. Assume finally $\beta > \gamma$. In this case, N_l is defined as

$$N_l \coloneqq \left\lceil 2\epsilon^{-2} c_2 \frac{m^{-(\gamma+\beta)\frac{l}{2}}}{1 - m^{-(\beta-\gamma)/2}} \right\rceil.$$

The variance can be estimated similarly to the previous case:

$$\sum_{l=0}^{L} \mathrm{var}[\widehat{Y}_l] \leq \frac{1}{2}\epsilon^2 \Big(1 - m^{-(\beta-\gamma)/2}\Big) \sum_{l=0}^{L} m^{-(\beta-\gamma)\frac{l}{2}} \leq \frac{\epsilon^2}{2}.$$

The cost can be estimated by

$$\mathrm{Cost}(\widehat{Q}^{\mathrm{ML}}_{L,\{N_l\}}) < c_3 \left(2\epsilon^{-2} c_2 \Big(1 - m^{-(\gamma-\beta)/2}\Big)^{-2} + \sum_{l=0}^{L} m^{\gamma l} \right).$$

Since $\epsilon < e^{-1} < 1$ and $\alpha \geq \frac{1}{2}\gamma$, we have $\epsilon^{-\gamma/\alpha} \leq \epsilon^{-2}$. Finally, using the inequality (2.14), we get

$$\mathrm{Cost}(\widehat{Q}^{\mathrm{ML}}_{L,\{N_l\}}) = O(\epsilon^{-2})$$

and the proof is finished.

□

Let us now state the version of the complexity theorem for the FE solution defined in Chapter 1. It is a result analogous to Theorem 2.7 for the standard MC method.

Theorem 2.11 (MLMC FEM complexity)**.** *Let f be sufficiently smooth so that Theorem 1.16 holds. Let $m \in \mathbb{N}, m > 1$ and let $h_0, h_1, \dots$ be discretisation parameters satisfying $h_0 > 0$ and $h_l = \frac{1}{m} h_{l-1}$. Assume that the cost per sample is given by $C_l = O(h_l^{-2})$. Then, for any $0 < \epsilon < \exp^{-1}$ there exist $L \in \mathbb{N}$ and $\{N_l\}_{l=0}^L$ such that*

$$\|\mathbb{E}[Q] - \widehat{Q}^{ML}_{L,\{N_l\}}\|^2_{L^2(\Omega,\mathbb{R})} < \epsilon^2$$

and

$$Cost(\widehat{Q}^{ML}_{L,\{N_l\}}) = O(\epsilon^{-2}),$$

where $\widehat{Q}^{ML}_{L,\{N_l\}}$ is the MLMC estimator for $\mathbb{E}[Q]$ in the sense of Definition 2.8 corresponding to the solution u_h of the discrete AVP with random data (Problem 1.13).

Proof. In order to prove the theorem, it suffices to verify the assumptions of Theorem 2.10. From Theorem 2.7 we know that $\alpha = \gamma = 2$. It thus remains to determine the value of β. To this end, let us estimate

$$\begin{aligned}
\operatorname{var}[Y_l] &= \mathbb{E}[Y_l^2] - \mathbb{E}[Y_l]^2 \\
&\leq \mathbb{E}[(Q_{h_l} - Q + Q - Q_{h_{l-1}})^2] \\
&\leq \mathbb{E}[(Q - Q_{h_l})^2] + 2\mathbb{E}[(Q_{h_l} - Q)(Q - Q_{h_{l-1}})] + \mathbb{E}[(Q - Q_{h_{l-1}})^2] \\
&\leq 2\mathbb{E}[(Q - Q_{h_l})^2] + 2\mathbb{E}[(Q - Q_{l-1})^2].
\end{aligned}$$

In the last inequlaity, we used the Cauchy–Schwarz inequality and the fact that $2ab \leq a^2 + b^2$ for $a, b \in \mathbb{R}$. The last expression can be estimated using Theorem 1.16 as

$$\begin{aligned}
&2\mathbb{E}[(Q - Q_{h_l})^2] + 2\mathbb{E}[(Q - Q_{h_{l-1}})^2] \\
&= 2\|G(u) - G(u_{h_l})\|^2_{L^2(\Omega,\mathbb{R})} + 2\|G(u) - G(u_{h_{l-1}})\|^2_{L^2(\Omega,\mathbb{R})} \\
&\leq 2C^2 h_l^4 + 2C^2 h_{l-1}^4 \\
&= 2C^2(1 + m^4) h_l^4 \\
&= C h_l^4,
\end{aligned}$$

where C is a generic constant. Thus the assumption of Theorem 2.10 holds with $\beta = 4$ and the proof is finished. □

In practice we use an adaptive algorithm to determine the values L and $\{N_l\}_{l=0}^L$. The value L has to be large enough in order to bound the bias error and the values N_l have to be large enough to bound the variance term; see the bias-variance decomposition (Lemma 2.9). What we need to estimate the MSE using the bias-variance decomposition are some computable error estimates, which are based on the sample average and sample variance. These will be the topic of the next section along with the adaptive algorithm.

2.3.2 Adaptive MLMC algorithm

As suggested above, the adaptive MLMC algorithm aims to compute the optimal values of L and $\{N_l\}_{l=0}^L$. It does so using sample averages $\widehat{Y}_l$ (see Definition 2.8) and sample variances

$$s_l^2 \coloneqq \frac{1}{N-1} \sum_{k=1}^{N_l} \left(Y_l^{(k)} - \widehat{Y}_l\right)^2 \tag{2.16}$$

of the random variables $\{Y_l\}_{l=0}^L$. What we also need in the algorithm is the cost per sample C_l which is given by the solver we use to solve the corresponding FE system.

We now describe how the adaptive algorithm computes the new values L and $\{N_l\}_{l=0}^L$. For this, we need the following lemma:

Lemma 2.12. *Assume that there exists $\tilde{h} > 0$ such that the function $h \mapsto |\mathbb{E}[Q_h - Q]|$ is strictly decreasing for all $h \leq \tilde{h}$ and satisfies*

$$\exists c_1, c_2 > 0 \, \forall h \leq \tilde{h} : c_1 h^\alpha \leq |\mathbb{E}[Q_h - Q]| \leq c_2 h^\alpha. \tag{2.17}$$

Further, let $l \in \mathbb{N}$ be such that $h_{l-1} \leq \tilde{h}$ and denote $r \coloneqq \frac{c_1}{c_2}$. If $rm^\alpha - 1 > 0$ then

$$|\mathbb{E}[Q_{h_l} - Q]| \leq \frac{1}{rm^\alpha - 1} |\mathbb{E}[Y_l]|,$$

where the notation is as in Theorem 2.10.

Proof. We employ the reverse triangle inequality to estimate

$$\Big| |\mathbb{E}[Q_{h_l} - Q]| - |\mathbb{E}[Q_{h_{l-1}} - Q]| \Big| \leq \Big| \mathbb{E}[Q_{h_l} - Q] - \mathbb{E}[Q_{h_{l-1}} - Q] \Big| = |\mathbb{E}[Y_l]|. \tag{2.18}$$

Since the error is strictly decreasing, the expression on the left-hand side can be rewritten as

$$\begin{aligned} \Big| |\mathbb{E}[Q_{h_l} - Q]| - |\mathbb{E}[Q_{h_{l-1}} - Q]| \Big| &= |\mathbb{E}[Q_{h_l} - Q]| \Big| 1 - \frac{|\mathbb{E}[Q_{h_{l-1}} - Q]|}{|\mathbb{E}[Q_{h_l} - Q]|} \Big| \\ &= |\mathbb{E}[Q_{h_l} - Q]| \Big(\frac{|\mathbb{E}[Q_{h_{l-1}} - Q]|}{|\mathbb{E}[Q_{h_l} - Q]|} - 1 \Big). \end{aligned} \tag{2.19}$$

Using the inequality (2.17), we obtain

$$\begin{aligned} |\mathbb{E}[Q_{h_l} - Q]| &\leq c_2 h_l^\alpha, \\ |\mathbb{E}[Q_{h_{l-1}} - Q]| &\geq c_1 h_{l-1}^\alpha. \end{aligned}$$

Using the fact that $h_{l-1} = m h_l$, the right-hand side of (2.19) can then be estimated from below as

$$\begin{aligned} |\mathbb{E}[Q_{h_l} - Q]| \Big(\frac{|\mathbb{E}[Q_{h_{l-1}} - Q]|}{|\mathbb{E}[Q_{h_l} - Q]|} - 1 \Big) &\geq |\mathbb{E}[Q_{h_l} - Q]| \Big(\frac{c_1}{c_2} \frac{h_{l-1}^\alpha}{h_l^\alpha} - 1 \Big) \\ &= |\mathbb{E}[Q_{h_{l-1}} - Q]| (rm^\alpha - 1). \end{aligned}$$

This, together with (2.18), yields

$$|\mathbb{E}[Q_{h_{l-1}} - Q]| \leq \frac{1}{rm^\alpha - 1} |\mathbb{E}[Y_l]|.$$

□

The lemma allows us to estimate the bias error using the sample average $\widehat{Y}_l$ as

$$|\mathbb{E}[Q_{h_l} - Q]| \leq \frac{1}{rm^\alpha - 1} |\mathbb{E}[Y_l]| \approx \frac{1}{rm^\alpha - 1} |\widehat{Y}_l|. \tag{2.20}$$

This allows us essentially to determine the value of L (if the bias error on the level L is not small enough, we increase L).

Let us now describe how to determine the (optimal) values $\{N_l\}_{l=0}^L$. To this end, we minimize the total computational cost of the MLMC estimator for a fixed variance. The computational cost of the MLMC estimator can be written as

$$\mathrm{Cost}(\widehat{Q}^{\mathrm{ML}}_{L,\{N_l\}}) = \sum_{l=0}^{L} N_l C_l, \tag{2.21}$$

where $C_l = \mathrm{Cost}(Y_l^{(k)})$ is the cost per one sample of Y_l as in the complexity theorem. We can now minimize (2.21) with respect to $N_l, l = 0 \ldots, L$ under the constraint

$$\sum_{l=0}^{L} \frac{V_l}{N_l} = \frac{\epsilon^2}{2}.$$

This constrained optimization problem can be solved by the method of Lagrange multipliers. A straightforward computation yields that the optimal values of N_l satisfy

$$N_l = \lambda \sqrt{\frac{V_l}{C_l}} \tag{2.22}$$

with an implied constant λ satisfying

$$\lambda = \frac{2}{\epsilon^2} \sum_{l=0}^{L} \sqrt{V_l C_l}.$$

We can now proceed to the adaptive algorithm itself. At each iteration, the algorithm estimates the optimal values $\{N_l\}_{l=0}^L$ using the sample variance (2.16) and the formula (2.22). Then it checks using (2.20) if the bias error is under the desired tolerance and if not, increases the value L by one and initializes N_L. If the bias error is under the given tolerance and so is the variance, estimated by $\sum_l s_l^2/N_l$ (see the bias-variance decomposition, Lemma 2.9), the algorithm terminates, otherwise the process is repeated.

We proceed to the formulation of the adaptive MLMC algorithm; see Algorithm 1 (the algorithm is adapted from [6]). In the following section we analyse in more detail the cost of the MLMC algorithm.

2.3.3 Cost analysis

This section aims to analyse the contributions of the individual levels in the MLMC method to the overall cost. The total cost is given by (2.21), i.e., the total cost is given by the sum of the costs on individual levels. Let us express how the total cost is increased per level. This is given by the ratio

$$\frac{C_{l+1} N_{l+1}}{C_l N_l}. \tag{2.23}$$

Assumptions (2.9) and (2.10) tell us that $\mathrm{var}[Y_l] = O(h_l^\beta)$ and $C_l = O(h_l^{-\gamma})$, respectively. If we moreover assume that $\mathrm{var}[Y_l] \simeq h_l^\beta$ and $C_l \simeq h_l^{-\gamma}$, where the

Algorithm 1: Adaptive MLMC algorithm

Input: h_0, m, ϵ, $L = 1$, $L_{\max}$, $N_0 = N_1 = N_{\text{init}}$

Output: $\widehat{Q}^{\text{ML}}_{L,\{N_l\}}$

while $L \leq L_{max}$ **do**

  for $l = 0$ **to** L **do**

    Compute N_l new samples $Y_l^{(k)}$ using Def. 2.8;

    Compute $\widehat{Y}_l$, s_l^2 and estimate C_l;

  end

  Update estimates for N_l using (2.22);

  if $|\widehat{Y}_L| > \frac{rm^{\alpha}-1}{\sqrt{2}}\epsilon$ **then**

    $L \coloneqq L + 1$;

    $N_L \coloneqq N_{\text{init}}$;

  if $|\widehat{Y}_L| \leq \frac{rm^{\alpha}-1}{\sqrt{2}}\epsilon$ *and* $\sum_{l=0}^{L} s_l^2/N_l \leq \epsilon^2/2$ **then**

    $\widehat{Q}^{\text{ML}}_{L,\{N_l\}} \coloneqq \sum_{l=0}^{L} \widehat{Y}_l$;

end

symbol $\simeq$ means "is equal up to a multiplicative constant", then we can rewrite (2.23) as

$$\frac{C_{l+1}N_{l+1}}{C_l N_l} = \sqrt{\frac{h_{l+1}^{\beta-\gamma}}{h_l^{\beta-\gamma}}} = \sqrt{\frac{(m^{-1}h_l)^{\beta-\gamma}}{h_l^{\beta-\gamma}}} = m^{\frac{\gamma-\beta}{2}},$$

where we used the assumption that $h_l = m^{-1}h_{l-1}$ (see Theorem 2.10). We observe that the contribution of each individual level to the overall cost depends on the values of β and γ. In general we can say that the cost per level is increased (decreased) by the factor $m^{(\gamma-\beta)/2}$. Based on the values of β and γ, three cases can be naturally distinguished which correspond to the three cases in Theorem 2.10. The most interesting for our future considerations is the case when $\beta > \gamma$ where the cost on the coarsest level dominates. We summarize our observations in the following corollary:

Corollary 2.13. *Let the assumptions of Theorem 2.10 hold and let* $var[Y_l] \simeq h_l^{\beta}$ *and* $C_l \simeq h_l^{-\gamma}$ *where* $\beta > \gamma$. *Then the cost* C_0N_0 *on the coarsest level dominates and it holds that*

$$\frac{C_{l+1}N_{l+1}}{C_l N_l} = m^{\frac{\gamma-\beta}{2}},$$

i.e., the cost per level decays with the factor $m^{\frac{\gamma-\beta}{2}}$.

3. Numerical linear algebra methods

In Chapter 1 we were concerned with the numerical solution of a PDE via the finite element method. Our focus in this chapter will be solving the resulting system of linear algebraic equations in finite precision arithmetic. The consequences of using finite precision arithmetic in Monte Carlo methods are then studied in Chapter 4.

3.1 Floating point arithmetic

This section gives a brief overview of principles of computations in floating point arithmetic and definitions of basic notions. A floating point (FP) number system $\mathbb{F}$ is a finite subset of real numbers whose elements can be written in a specific form. The FP system $\mathbb{F}$ is characterised by three parameters, the so-called significand (mantissa), base, and exponent. One quantity determined by these parameters is the so-called range of $\mathbb{F}$. The range is, roughly speaking, a subset of the real numbers representable in $\mathbb{F}$. The range of $\mathbb{F}$ consists of two parts: The positive part is the interval between the smallest positive FP number and the largest FP number. The negative part is defined analogously. A thorough description of $\mathbb{F}$ can be found in Chapter 2 of [19]; we will focus here only on the properties crucial for our work.

An important property of the FP system $\mathbb{F}$ is the machine precision ϵ_M. It is the distance between the number 1 and next floating point number. Another important property is the unit roundoff δ, which is defined as $\delta := \frac{1}{2}\epsilon_M$. In the work [19] the unit roundoff is denoted by the lowercase u. We denote it here by δ to distinguish it from a solution of a PDE denoted in this work by u and the tolerance of the algorithms denoted in Chapter 2 by ϵ. The process which assigns an element $fl(x) \in \mathbb{F}$ to an element $x \in \mathbb{R}$ is called rounding. The important property is that any element lying in the range of $\mathbb{F}$ can be approximated within $\mathbb{F}$ via rounding with a relative error not larger than δ. For details see Theorem 2.2 in [19].

We assume that all computations which are carried out in FP arithmetic are carried out under following standard model. We assume that for all $x, y \in \mathbb{F}$ we have

$$fl(x \operatorname{op} y) = (1+\mu)(x \operatorname{op} y), \quad |\mu| \leq \delta, \quad \operatorname{op} = +, -, \times, /. \tag{3.1}$$

Note that this assumption implies that we assume all of the considered quantities to lie within the range of $\mathbb{F}$. The standard model is valid in particular for IEEE arithmetic. IEEE arithmetic is a technical standard of floating point arithmetic which assumes all of the preliminaries above and adds other technical assumptions; see [19] for an overview. The IEEE standard defines several basic formats. In this work we use formats both standardised and not standardised by IEEE. All of the formats we will use are hardware-supported, namely by the NVIDIA H100 SXM GPU (see Section 3.4). To be specific, in this thesis we use quarter (q43), half, single, and double precision. Note that the base of all of these formats is 2.

	Unit roundoff	Range	Bits
quarter (q52)	$2^{-3} \approx 1.3 \times 10^{-1}$	$10^{\pm 5}$	8
quarter (q43)	$2^{-4} \approx 6.3 \times 10^{-2}$	$10^{\pm 2}$	8
half	$2^{-11} \approx 4.9 \times 10^{-4}$	$10^{\pm 5}$	16
bfloat16	$2^{-8} \approx 3.9 \times 10^{-3}$	$10^{\pm 38}$	16
single	$2^{-24} \approx 6.0 \times 10^{-8}$	$10^{\pm 38}$	32
double	$2^{-53} \approx 1.1 \times 10^{-16}$	$10^{\pm 308}$	64

Table 3.1: Range, value of the unit roundoff, and number of bits required for storing one number for various precision formats. Not all of them are defined by the IEEE standard.

The corresponding values of the unit roundoff are denoted by δ_q δ_h, δ_s, and δ_d, respectively, and their values are shown in Table 3.1 along with the range and the number of bits the number occupies in computer memory. Two different formats of quarter precision are shown — q43 and q52. The former has 4 exponent bits, 3 significand bits, whereas the latter has 5 exponent bits, 2 significand bits. When an algorithm is run, all the computations are assumed to be carried out under the standard model (3.1). This assumption allows us to analyse the error of the algorithm. For simplicity, we will often abbreviate "the format of the floating point arithmetic with unit roundoff δ" to "the precision δ". The results of the error analysis of the algorithms important for this work are given in the following sections. We will be interested not only in the error analysis of the algorithms but also in their computational complexity (cost in terms of FLOPs).

3.2 Cholesky factorization

As discussed in Section 1.2.3, seeking the FE solution of our elliptic boundary value problem is equivalent to solving a system of linear algebraic equations. As pointed out, the matrix of this system has some special properties which have to be taken into account when the system is solved, namely, the matrix is sparse, symmetric, and positive definite. One prominent class of methods suitable for solving systems with such a matrix are sparse direct solvers based on Cholesky factorization, which will be our focus in this section. Here we follow [19].

3.2.1 Basic properties and computation

The Cholesky factorization is a matrix factorization of the form $A = R^T R$, where R is upper triangular with positive diagonal elements. The following theorem guarantees its existence and uniqueness for a symmetric positive definite matrix.

Theorem 3.1. *Let $A \in \mathbb{R}^{n\times n}$ be a symmetric positive definite matrix. Then there exists a unique upper triangular matrix $R \in \mathbb{R}^{n\times n}$ with positive diagonal elements such that $A = R^T R$.*

Proof. See [19], Theorem 10.1. □

Algorithm 2: Column Cholesky factorization

Input: $A \in \mathbb{R}^{n\times n}$ symmetric positive definite

Output: Upper triangular $R \in \mathbb{R}^{n\times n}$ with positive diagonal entries satisfying $A = R^T R$

for $j = 1$ **to** n **do**
  for $i = 1$ **to** $j - 1$ **do**
    $r_{i,j} = (a_{i,j} - \sum_{k=1}^{i-1} r_{k,i} r_{k,j})/r_{i,i}$;
  end
  $r_{j,j} = (a_{j,j} - \sum_{k=1}^{j-1} r_{k,j}^2)^{1/2}$;
end

We now turn our attention to how to compute the Cholesky decomposition. For a dense matrix, we have three basic options: To build the matrix R column by column, row by row, or the so-called submatrix factorization. The standard column algorithm will be described here. It can be straightforwardly derived from the equation $A = R^T R$:

Consider $A = (a_{i,j})_{i,j=1}^n$ and $R = (r_{i,j})_{i,j=1}^n$. Let us now rewrite the equation $A = R^T R$ element by element:

$$a_{i,j} = \sum_{k=1}^{i} r_{k,i} r_{k,j}, \quad j \geq i.$$

Solving the equation corresponding to $a_{1,1}$ gives us now $r_{1,1}$. Consequently, solving the equation corresponding to $a_{1,2}$ yields $r_{1,2}$ and in a similar way we can obtain $r_{2,2}$, $r_{1,3}$, $r_{2,3}$, $r_{3,3}$, etc. Hence we obtain the matrix R column by column, which can be written down in the form of an algorithm; see Algorithm 2, taken from [19], Algorithm 10.2. The computational complexity of the algorithm is $O(n^3)$ FLOPs.

The Cholesky factorization can be used to solve a system $Ax = b$: First we decompose $A = R^T R$ to obtain the system $R^T Rx = b$. Denoting $y := Rx$, we can solve the system $R^T y = b$ using forward substitution. Finally, the system $Rx = y$ is solved by backward substitution. The complexity of solving the system $Ax = b$ using Algorithm 2 is $O(n^3)$.

The complexity of Cholesky factorization can be, in our case, reduced significantly by exploiting the sparsity of the matrix A. We will not go into details here; only a brief overview of the complexity results will be given. There are results showing that for the class of matrices coming from the discretisation of a PDE via the conforming FEM in 2D (which is our case), the complexity of the Cholesky factorization can be reduced from $O(n^3)$ to $O(n^{3/2})$. The complexity of the substitution part can be in this case reduced from $O(n^2)$ to $O(n \log n)$; see [27], Section 2.2. Such a class of techniques for sparse Cholesky factorization is implemented in the library CHOLMOD used in Matlab.

In this work we use another variant of the Cholesky factorization - the $LDL^\top$ factorization, since we can avoid computing the square roots on the diagonal, see [19], Chapter 10, for a reference. The results of the error analysis of the $LDL^\top$ factorization are analogous to the standard Cholesky factorization presented in this section, according to [19]. For the numerical results, it is not of high importance which of these two algorithms we choose.

3.2.2 Summary of error analysis

As suggested in Section 3.1, in practice Algorithm 2 is run in floating point arithmetic and only an approximation of the true factorization is computed. The Cholesky factorization is used to solve a system $Ax = b$ as suggested in Section 3.2.1 and we are interested in the error of the computed solution.

Assume that $\widehat{x}$ is the solution of the system $Ax = b$ in a FP arithmetic format with the unit roundoff δ. This means in our context that the Cholesky factorization and forward and backward substitution are all performed in this FP arithmetic format and as a result, they produce the solution $\widehat{x}$. The error of the approximation can be studied from many perspectives. One possibility is, for instance, to analyse the absolute forward error $\|x - \widehat{x}\|$ or the relative forward error $\|x - \widehat{x}\|/\|x\|$ in a suitable norm (or componentwise). We will be interested in the so-called backward error which will prove useful later. The backward error aims to express the approximate solution $\widehat{x}$ as the true solution to a problem with perturbed input data. Let us first introduce an auxiliary notation. Let $n \in \mathbb{N}$ be such that $n\delta < 1$. We denote

$$\gamma_n \coloneqq \frac{n\delta}{1 - n\delta}.$$

The main result of the backward error analysis is given by the following theorem adapted from [19], Theorem 10.4.

Theorem 3.2. *Let $A \in \mathbb{R}^{n\times n}$ be symmetric positive definite and suppose Cholesky factorization produces a computed factor $\widehat{R}$ and a computed solution $\widehat{x}$ to $Ax = b$. If $\max\Big((3n + 1)\delta, n\gamma_{n+1}\Big) < 1/2$ then*

$$(A + \Delta A)\widehat{x} = b, \quad \|\Delta A\|_2 \leq 4n(3n + 1)\delta\|A\|_2, \tag{3.2}$$

where $\|\cdot\|_2$ denotes the spectral norm of a matrix.

This theorem tells us that this algorithm for solving a system of linear equations is backward stable. In practice it turns out that the estimate (3.2) often overestimates the true error and thus this result may be useful only theoretically.

For the purpose of the error analysis in Theorem 3.2 we assumed that Cholesky factorization applied to the matrix A succeeds. Wilkinson showed that Cholesky factorization is guaranteed to succeed if $20n^{3/2}\kappa_2(A)\delta \leq 1$, where $\kappa_2(A)$ denotes the condition number of A (with respect to the spectral norm); see [19], Section 10.1.1. There are further results which guarantee success of the factorization under certain conditions on the singular values of A; see Theorem 10.7 in [19].

Not all arithmetic operations in the algorithm for solving $Ax = b$ have to be carried out in one precision. There are techniques which allow us to improve the accuracy of the computed solution via, e.g., iterative refinement, which will be the topic of the following section. This and the fact that the theoretical estimates often exaggerate the true error motivate us to introduce the so-called effective precision δ_e. The effective precision is not necessarily a hardware or software-supported precision (like the IEEE standards, e.g.). It is rather a parameter expressing how accurately the solution to $Ax = b$ is actually computed.

Let $\widehat{x}$ be a solution of $Ax = b$ computed by an algorithm. The solution is said to be computed effectively to precision δ_e if

$$\frac{\|b - A\widehat{x}\|_{\mathbb{R}^n}}{\|b\|_{\mathbb{R}^n}} \leq C\delta_e \tag{3.3}$$

for a constant $C > 0$. Here $\|\cdot\|_{\mathbb{R}^n}$ denotes the Euclidean norm, although it is possible to use any other norm in principle. The constant C may or may not be dependent on the matrix A and other input data; this is problem-dependent. As we can see, this definition is very broad and the effective precision is not uniquely defined; the property can be satisfied by more than one value.

3.3 Iterative refinement

Iterative refinement is a technique used to enhance the accuracy of a numerical solution to a linear system; see [19], Chapter 12, for an overview. The general scheme of iterative refinement involves computing a residual vector, which represents the difference between the exact solution and the current approximate solution, and then applying a correction to the current approximation to reduce the error. In this way, the process can be repeated iteratively until the limiting level of accuracy is achieved.

As explained in [27], the first use of iterative refinement as well as the first rounding error analysis are attributed to Wilkinsion, who made use of the fact that the inner product (and thus the residual) could be computed at twice the working precision on many computers of that period with no additional cost. The first error analysis in floating point arithmetic is attributed to Moler and can be viewed as a foundation for the recent analyses of iterative refinement. Today, iterative refinement is a widely used technique to improve the forward error of a solution, to recover the stability of the solver, and to accelerate the solution of a linear system using low precision arithmetic; see [1]. In this work, the use of iterative refinement is motivated by the cost reduction, but not only: it turns out that we do not often require that the solution is recovered to full accuracy. Iterative refinement allows us to factorize the matrix cheaply using low precision and then cheaply improve the accuracy of the computed solution to the required level in the sense of the relative residual.

The particular version of iterative refinement which we use in this work is adapted from [5]. Consider a linear system $Ax = b$ where $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$. The iterative refinement algorithm contains explicitly three precisions, δ_r, δ, and δ_f, and one precision, δ_s, implicitly. Note that in this section the symbol δ_s does not denote single precision and we use it here to keep the indices in accordance with the aforementioned paper, where the precision are denoted by u_r, u, u_f and u_s, respectively. The definitions of these precisions are, as explained in [5]:

- "δ is the precision at which the data A, b and the solution x are stored (the working precision),
- δ_f is the precision at which the factorization of A is computed,
- δ_r is the precision at which residuals are computed,
- δ_s is the precision at which the correction equation is (effectively) solved."

All of the precisions δ_r, δ, δ_s and δ_f will take in the cases of interest one of the values of quarter, half, single, or double precision and it is assumed $\delta_r \leq \delta \leq \delta_s \leq \delta_f$. We proceed directly to the algorithm which is a special case of Algorithm 1.1

Algorithm 3: Iterative refinement

Input: $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$, both in precision δ, tolerance $TOL > 0$.
Output: an approximation x_i of the solution x stored in precision δ.

1 Factorize A in precision δ_f;
2 Solve $Ax_0 = b$ in precision δ_f by substitution and store x_0 at precision δ;
3 **for** $i = 0$ **to** i_{max} **do**
4 Compute $r_i = b - Ax_i$ at precision δ_r and round r_i to precision δ_s;
5 **if** $\|r_i\|/\|b\| < TOL$ **then**
6 Exit algorithm;
7 Solve $Ad_i = r_i$ by substitution at precision δ_f or δ using the factorization from step 1 and store d_i at precision δ;
8 $x_{i+1} = x_i + d_i$ at precision δ;
9 **end**

from [5]; see Algorithm 3. As mentioned above, δ_s is the precision at which the equation in step 7 is solved. In our case the precision δ_s takes the value of δ_f, since the factorization is carried out in precision δ_f and we always consider the factorization method to be either Cholesky or $LDL^\top$ decomposition.

The precision to be used in step 7 of the algorithm is always specified when iterative refinement is used. By using precision δ in step 7 of the algorithm, no numerical benefit from the perspective of the limiting accuracy can be obtained. However, we benefit from it in the sense that, roughly speaking, a higher variety of accuracies can be achieved, see Section 5.1. There is also no point in using extra precision to compute the residual in step 4 of the algorithm, since we are interested only in bounding the backward error; see [5].

It is worth noting that the stopping criterion in Algorithm 3 given by the relative residual norm is non-standard and is motivated by our specific use of iterative refinement; see Chapters 4 and 5. The key idea is the following: The tolerance for the relative residual norm in step 5 of the algorithm corresponds to the value δ_e from the previous section. That is, if we set in step 5 the tolerance $TOL \coloneqq \delta_e$ and Algorithm 3 converges, then the produced solution is computed effectively to precision δ_e in the sense of (3.3). For a different and more standard choice of the stopping criteria, see [8]. The convergence of Algorithm 3 is in [5] analysed thoroughly. We give here only a brief overview of the results which are of our interest. We are particularly interested in the normwise backward error analysis in the $\|\cdot\|_{\mathbb{R}^n}$ norm (the Euclidean norm). The normwise backward error analysis is presented in [5] in the $\|\cdot\|_\infty$ norm. An analogous result can be obtained for the $\|\cdot\|_{\mathbb{R}^n}$ norm and the proof can be the same step by step. The following lemma summarizes the convergence conditions along with the convergence results. It is a direct consequence of Corollary 4.2 in [5].

Lemma 3.3. *Let Algorithm 3 without steps 5 and 6 be applied to a linear system $Ax = b$ with a nonsingular matrix $A \in \mathbb{R}^{n\times n}$. Let the following conditions hold:*

1. *The solver used in step 7 produces a computed solution $\widehat{d}_i$ to $Ad_i = \widehat{r}_i$ satisfying*

$$\|\widehat{r}_i - A\widehat{d}_i\|_{\mathbb{R}^n} \leq \delta_s(c_1\|A\|_2\|\widehat{d}_i\|_{\mathbb{R}^n} + c_2\|\widehat{r}_i\|_{\mathbb{R}^n}),$$

where $c_1, c_2 > 0$ and

2. *the quantity* $\phi \coloneqq (c_1\kappa_2(A) + c_2)$, *where* c_1 *and* c_2 *are the same as above, is sufficiently smaller than* 1.

Here $\kappa_2(A)$ *denotes the condition number of* A *(with respect to the spectral norm). Under these assumptions the residual is reduced in each iteration by a factor approximately* ϕ *and the algorithm eventually produces a computed solution* $\widehat{x}_i$ *backward stable to the working precision, i.e.,*

$$\|b - A\widehat{x}_i\|_{\mathbb{R}^n} \lesssim p\delta(\|b\|_{\mathbb{R}^n} + \|A\|_2\|\widehat{x}_i\|_{\mathbb{R}^n}), \tag{3.4}$$

where p *denotes the maximum number of nonzeros in any row of the augmented matrix* $[A\, b]$. *The symbol* $\lesssim$ *means "is less then or equal up to a multiplicative constant".*

For the matrix factorization we use the Cholesky factorization, which leads to the following estimate for the constant c_1 from Lemma 3.3 (see Theorem 3.2):

$$c_1 \leq 4n(3n + 1).$$

This lemma has some practical consequences. Under the assumptions of the lemma, the convergence of the algorithm is monotonic in terms of the norm of the residual and the algorithm terminates using the termination condition given in steps 5 and 6 as long as the tolerance is not too small. The limiting value of the tolerance can be deduced from (3.4).

In this work we use Algorithm 3 extensively to achieve the desired accuracy of the computed solution. We obtain additional benefits from using the iterative refinement in terms of the cost. The point is to reuse the factorization computed in step 1 of the algorithm in step 7 of the algorithm. If we reuse the factorization as suggested, we can improve the accuracy of the solution with a small additional cost, because the factorization is typically more costly than the substitution part in terms of the required number of operations, see Section 3.2.1.

3.3.1 Iterative refinement for FE linear systems

In this section we aim to put iterative refinement into the theoretical framework of our work, namely that of Chapter 1. We investigate the behaviour of iterative refinement applied to linear systems stemming from the finite element method. For simplicity, we shall assume that the underlying PDE does not contain any randomness, i.e., we are in the setting of Problem 1.1, and we wish to solve the corresponding FE system (1.6). Note that our considerations in this section are restricted to the case $D \subset \mathbb{R}^2$ but they can be straightforwardly generalised to cases when the domain D is a subset of $\mathbb{R}^n$, $n \neq 2$, and when the PDE is of a different order (different than 2).

Consider the FE system (1.6) corresponding to the discretisation parameter $h > 0$. Our goal is to reformulate Lemma 3.3 in terms of the data of the original problem (Problem 1.5). To this end we need some auxiliary inequalities. These inequalities, which we summarize here, can be found in [10].

Lemma 3.4. *Let* $Ax = b$ *be the FE system corresponding to Problem 1.5. Then the following estimates hold:*

1. $\|A\|_2 \leq c$,

2. $\kappa_2(A) \leq ch^{-2}$, *and*

3. $\|b\|_{\mathbb{R}^n} \leq ch\|f\|_{L^2(D)}$,

where κ_2 is the condition number of A with respect to the spectral norm and the generic constant c is independent of the discretisation parameter h.

Proof. Note that all theorems referenced in this proof are from [10] and more details can be found there. The first claim follows from the first part of the proof of Theorem 9.11 (the last inequality in the first part) and Theorem 9.8. The inequality from Theorem 9.11 gives us a bound on $\|A\|_2$ using h and the eigenvalues of the mass matrices. Theorem 9.8 gives us the bounds for the eigenvalues of the mass matrices.

The second point follows from Theorem 9.11 and Example 9.13, since in our case $s = t = 1$ in Theorem 9.11.

The third point can be proved as follows. Let us define the mass matrix as in [10], page 386, i.e.,

$$\mathcal{M}_{i,j} = \Big(\int_D \phi_i \phi_j\Big),$$

$i, j = 1, \ldots, n$, where ϕ_i, ϕ_j are the basis functions described in Section 1.2.3. Let $y \in \mathbb{R}^n$. Denote by v_h the corresponding element of $V_{h,0}$ represented by y, i.e.,

$$v_h = \sum_{j=1}^{n} y_j \phi_j. \tag{3.5}$$

According to (9.5) in [10], we have for all $v_h \in V_{h,0}$

$$\mu_{min}^{1/2} \|y\|_{\mathbb{R}^n} \leq \|v_h\|_{L^2(D)} \leq \mu_{min}^{1/2} \|y\|_{\mathbb{R}^n}, \tag{3.6}$$

where μ_{min} and μ_{max} denote the smallest and largest eigenvalue of $\mathcal{M}$, respectively. Further, from Theorem 9.8 in [10] we obtain that for any eigenvalue μ of $\mathcal{M}$ the following inequality holds:

$$c_1 h^2 \leq \mu \leq c_2 h^2, \tag{3.7}$$

where c_1 and c_2 are independent of h. Recall that the right-hand side b is defined for $i = 1, \ldots, n$ by $b_i = l(\phi_i)$, where l is from Problem 1.5. For any $y \in \mathbb{R}^n$ we obtain, using (3.5), (3.6), (3.7) and Hölder's inequality,

$$\begin{aligned}
\|y\|_{\mathbb{R}^n} &= \sup_{y \in \mathbb{R}^n} \frac{(b, y)}{\|y\|_{\mathbb{R}^n}} \\
&= \mu_{min}^{1/2} \sup_{v_h \in V_{h,0}} \frac{l(v_h)}{\|v_h\|_{L^2(D)}} \\
&\leq ch \sup_{v_h \in V_{h,0}} \frac{\|f\|_{L^2(D)} \|v_h\|_{L^2(D)}}{\|v_h\|_{L^2(D)}} \\
&= ch\|f\|_{L^2(D)},
\end{aligned}$$

which completes the proof. □

As a consequence of Lemma 3.4 and Lemma 3.3 we formulate the following corollary about the behaviour of iterative refinement for the FE systems.

Corollary 3.5. *Let $Ax = b$ be the FE system corresponding to Problem 1.5 and let the assumptions of Lemma 3.3 hold. Let ϕ and $\widehat{x}_i$ be as in Lemma 3.3. Then*

1. *the factor ϕ satisfies $\phi \leq kh^{-2} + c_2$ and*
2. *$\|b - A\widehat{x}_i\|_{\mathbb{R}^n} \lesssim p\delta(h\|f\|_{L^2(D)} + \|\widehat{x}_i\|_{\mathbb{R}^n})$,*

where c_2 is from Lemma 3.3 and $k = cc_1$ where c_1 is from Lemma 3.3 and c is from Lemma 3.4, part 1.

Proof. The proof follows immediately from Lemma 3.3 using the inequalities from Lemma 3.4. □

This lemma has some practical consequences. In the case of the MLMC method, we solve our FE problem on a hierarchy of meshes assuming that the meshes are uniformly refined by the factor $m > 1$; see Section 2.3. This lemma helps us to understand the behaviour of iterative refinement in this case, which will be useful later. If the assumptions of Corollary 3.5 are satisfied, we can conclude that the more we refine the mesh, the slower the iterative refinement converges (for a fixed value of δ_s) and the approximate rate is given by point 1. On the other hand, the limiting precision bounding the norm of the residual does not change dramatically as the mesh is refined according to point 2. This is a consequence of the fact that refining the mesh increases the condition number significantly, but does not drastically increase the norm of the matrix or right-hand side.

3.4 Performance of finite precision computations

The use of high-precision floating point arithmetic, e.g., double precision, in scientific computing has been standard practice for many years. However, as the size and complexity of scientific simulations increase, so does the need for more efficient computational resources. In recent years, there has been growing interest in the use of low precision floating point arithmetic to accelerate scientific computations while maintaining acceptable levels of accuracy. Recent advances in hardware and software have made it a viable option for many scientific applications. This section aims to discuss the performance of the discussed algorithms on current architectures and assess the gain stemming from using lower precision arithmetic. We also give a brief overview of some recent results regarding the performance gain from using lower precision compared to higher precision. This is used to make reasonable assumptions regarding the theoretical speedup of the algorithms enabled by the use of lower precision arithmetic.

As discussed in [3], the computational time of a numerical algorithm is, on current architectures, usually bounded by one of the following:

- Compute throughput, that is, the number of arithmetic operations that can be performed per cycle,

- memory bandwidth, that is, the number of operands than can be moved between levels of a memory hierarchy (sequential case) or between processors (parallel case) each cycle, or
- latency, which is the time from initiating a data request to having the data available for another instruction.

Note that in further considerations, we discuss sequential architectures. Considering the simplest model, we can assume that the number of arithmetic operations that can be performed per cycle is inversely proportional to the number of bits required to store a floating point number. Let us denote by T_d the computational time of a numerical algorithm run in double precision and denote in the same way T_s, T_h, and T_q. If the bounding quantity of the computational time is the compute throughput or the memory bandwidth, we can conclude that it holds that

$$\begin{aligned} T_s &= \frac{1}{2}T_d, \\ T_h &= \frac{1}{4}T_d, \\ T_q &= \frac{1}{8}T_d. \end{aligned} \tag{3.8}$$

This means that the computational time is proportional to the number of bits required to store one floating point number. In the rest of the thesis we assume (3.8) and we use this assumption to estimate the theoretical speedup of the algorithms. Since our assumptions on the computational model are very restrictive and this conclusion might not be valid in practice, we refer to some recent numerical results to verify (3.8) empirically.

Let us compare the computational performance using quarter, half, single, and double precision, denoted FP8, FP16, FP32, and FP64, respectively, on the NVIDIA H100 SXM5 GPU. The performance is given in terms of the number of floating point operations per second (FLOPS). The results, obtained from the NVIDIA specifications [24], are summarised in Table 3.2. Let us make further comments on the presented values. Some of the values of the table have been obtained using so-called tensor cores (TC). Tensor cores are specialised cores which enable the GPU to compute $D = AB + C$, where all matrices A, B, and C are of the dimension 4×4, in mixed precision with no loss of accuracy and a rapid acceleration of the computation; see [16]. However, not all of the considered precisions are TC-supported. Namely, of those shown in the table, single precision (FP32) is not TC-supported on NVIDIA H100 SXM5 GPU. On the contrary, quarter precision (FP8) is only supported using tensor cores. Note that further speedup (usually double) can be obtained if we assume certain sparsity of the data; see [24]. We, however, do not make this assumption and also the values displayed in Table 3.2 do not assume the sparsity. We observe that the use of quarter precision increased the performance remarkably more than $29\times$ compared to double precision (both using TC) which is much better than our presumption (3.8). This is, however, only the theoretical maximum which is attainable using this hardware. The practical performance gain depends on various aspects of the problem as well as the implementation. To estimate the gain realistically, we refer to the work [16].

	H100	V100
Peak FP64	33.5	6.8
Peak FP64 TC	66.9	-
Peak FP32	66.9	14
Peak FP16	133.8	28
Peak FP16 TC	989.4	85
Peak FP8 TC	1978.9	-

Table 3.2: Performance of NVIDIA H100 SXM5 and NVIDIA V100 PCIe GPUs for various precisions in TFLOPS (FLOPS means "floating point operations per second"). No sparsity is assumed.

In [16] several numerical experiments measuring the performance gain of low precision arithmetic were conducted using an NVIDIA V100 PCIe GPU. The theoretical peak performance of the NVIDIA V100 PCIe GPU is given in Table 3.2. In comparison with the NVIDIA H100 GPU, the peak performance of the NVIDIA V100 is not as high and some of the precisions, namely FP8 TC and FP64 TC, are not supported. In the work [16], the performance of the LAPACK algorithm GETRF for computing the LU factorization of a matrix was studied in various precisions. Their implementation computing the LU factorization accurately to half precision (using TC) achieves a speedup of $4\times$ to $5\times$ compared to double precision and a $2\times$ speedup over single precision (both not using TC). These results, which can be found in Section V of the aforementioned study, are in accordance with our assumption (3.8) about the computational time and confirm that this assumption is realistic. It is worth mentioning, however, that this practical result does not correspond completely to our case, since the latency, mentioned at the beginning of this section, comes into play.

4. Mixed precision Monte Carlo methods

In this chapter we present a thorough convergence analysis of the Monte Carlo methods discussed in Chapter 2 in finite precision arithmetic. Based on this analysis, we exploit low precision arithmetic to speed-up the Monte Carlo algorithms. We propose a mixed precision MLMC (MPMLMC) method designed to achieve accurate and reliable results while also minimizing computational costs. By utilizing the multilevel scheme, we can optimize the use of low precision arithmetic in the areas where it is sufficient, while still being able to switch to high-precision arithmetic in regions where it is necessary to maintain accuracy.

To our knowledge, this approach is new in the context of MC methods for PDE-based problems. Our work can be compared to the paper [4] where the authors discuss the idea of MPMLMC in the context of stochastic differential equations and then suggest a heuristic to determine a suitable precision for each level of the computation. Our approach is original in many ways; namely we

- provide a rigorous analysis of Monte Carlo methods in finite precision arithmetic via error estimates, namely of the standard MC method and the MLMC method;
- propose a novel adaptive mixed precision MLMC (MPMLMC) algorithm, determining the optimal precision on each level of discretisation using the theoretical error estimates with no additional cost;
- provide the theoretical background for applying the adaptive algorithm to the model elliptic PDE with random coefficients and a random right-hand side.

The correctness of the adaptive algorithm is demonstrated on numerous experiments in Chapter 5. While we demonstrate the proposed methods here on the model Problem 2.1, they can be straightforwardly generalised to tackle various types of problems.

4.1 Finite precision MC method

Throughout this chapter we will use the symbol $\widehat{u}_h$ to denote the solution of the discrete AVP with random data (Problem 1.13) computed effectively to precision δ. This means that

$$\widehat{u}_h(\cdot,\omega) = \sum_{j=1}^{n} \widehat{x}_j(\omega)\phi_j, \tag{4.1}$$

where $\widehat{x}$ is such that

$$\frac{\|b(\omega) - A(\omega)\widehat{x}(\omega)\|_{\mathbb{R}^n}}{\|b(\omega)\|_{\mathbb{R}^n}} \leq C\delta \tag{4.2}$$

and $C > 0$ is independent of the problem data and ω. The validity of this assumption in practice is discussed at the end of Section 4.2.4. See Section 1.3.4 for a reminder about the FEM in finite precision arithmetic and Section 3.2.2

for the definition of the effective precision. In principle, it is possible to use any discrete solution computed numerically in such a way that (4.1) and (4.2) are satisfied, not only the FE approximation. We proceed to the definition of the finite precision Monte Carlo (FPMC) estimator.

Definition 4.1 (Finite precision Monte Carlo finite element estimator)**.** *Let $\widehat{u}_h : D \times \Omega \to \mathbb{R}$ be as in* (4.1)*. Let $\mathbb{E}[Q]$ be the quantity of interest defined in Problem 2.1. Let $Q_{h,\delta}$ be a random variable defined as $Q_{h,\delta} : \Omega \to \mathbb{R}$, $\omega \mapsto G(\widehat{u}_h(\cdot, \omega))$. Define the finite precision Monte Carlo estimator (FPMC) for $E[Q]$ by*

$$\widehat{Q}_{h,N,\delta} \coloneqq \frac{1}{N} \sum_{k=1}^{N} Q_{h,\delta}^{(k)},$$

where $Q_{h,\delta}^{(k)}$ are random samples from $Q_{h,\delta}$.

This definition assumes that finite precision comes into play only when the FE system $Ax = b$ is solved. Once the system is solved, the estimate itself is computed using exact arithmetic. This is a reasonable assumption, since the cost of computing the estimate itself is negligible relative to the cost of solving the linear system N times. We now state the bias-variance decomposition, which is completely analogous to Theorem 2.5.

Lemma 4.2 (Bias-variance decomposition of the MSE of the FPMC estimator)**.** *The mean squared error of the FPMC estimator $\widehat{Q}_{h,N,\delta}$ from Definition 4.1 can be expanded as*

$$\mathbb{E}\Big[(\mathbb{E}[Q] - \widehat{Q}_{h,N,\delta})^2\Big] = (\mathbb{E}[Q - Q_{h,\delta}])^2 + \frac{var[Q_{h,\delta}]}{N}.$$

Proof. The proof is identical to the proof of Theorem 2.5. □

We proceed directly to the error estimate for the FPMC method. It is again general in the sense that the discrete solution does not have to be the FE solution defined in Chapter 1; it can be any discrete solution satisfying certain assumptions.

Theorem 4.3 (Error of the FPMC method)**.** *Assume that there exists α such that*

$$|\mathbb{E}[Q_{h,\delta} - Q]| = O(h^\alpha + \delta) \quad \text{for } h, \delta \to 0, \tag{4.3}$$

$$var[Q_{h,\delta}] = var[Q_h] + O(\delta) \quad \text{for } \delta \to 0, \tag{4.4}$$

where $Q_{h,\delta}$ is from Definition 4.1. Assume further there exists σ^2 such that for all h sufficiently small it holds that $var[Q_h] \leq \sigma^2$. Then for any $N \in \mathbb{N}$, $\delta > 0$, and $h > 0$, the error of the corresponding FPMC estimator $\widehat{Q}_{h,N,\delta}$ satisfies

$$\mathbb{E}\Big[(\mathbb{E}[Q] - \widehat{Q}_{h,N,\delta})^2\Big] \leq C\left(\left(h^{2\alpha} + \frac{\sigma^2}{N}\right) + \left(h^\alpha \delta + \delta^2 + \frac{\delta}{N}\right)\right).$$

Proof. Using the bias-variance decomposition (Lemma 4.2) we obtain

$$\begin{aligned}\mathbb{E}\Big[(\mathbb{E}[Q]-\widehat{Q}_{h,N,\delta})^2\Big] &= (\mathbb{E}[Q-Q_{h,\delta}])^2+\frac{\operatorname{var}[Q_{h,\delta}]}{N}\\ &\le C\bigg((h^\alpha+\delta)^2+\frac{\operatorname{var}[Q_h]+\delta}{N}\bigg)\\ &\le C\bigg(\Big(h^{2\alpha}+\frac{\sigma^2}{N}\Big)+\Big(h^\alpha\delta+\delta^2+\frac{\delta}{N}\Big)\bigg),\end{aligned}$$

where $C>0$ is a generic constant. □

Observe that if we add the assumption $\delta=O(h^\alpha)$ to the statement of Theorem 4.3 and add an assumption to bound the variance then the theorem can be further simplified.

Corollary 4.4 (Error of the FPMC method 2)**.** *Let the FPMC estimator $\widehat{Q}_{h,N,\delta}$ satisfy the assumptions of Theorem 4.3. Let $\widehat{Q}_{h,N}$ be the standard MC estimator using the same values of h and N as the FPMC estimator. Assume that the MC estimator $\widehat{Q}_{h,N}$ satisfies* (2.1)*, i.e., we have bias decay of order $O(h^\alpha)$. If we assume $\delta=O(h^\alpha)$ and if there exists $\sigma^2>0$ such that $var[Q_h]\le\sigma^2$ and $var[Q_{h,\delta}]\le\sigma^2$ then the mean squared errors of both estimators can be bounded by*

$$\left.\begin{aligned}&\mathbb{E}\Big[(\mathbb{E}[Q]-\widehat{Q}_{h,N,\delta})^2\Big]\\ &\mathbb{E}\Big[(\mathbb{E}[Q]-\widehat{Q}_{h,N})^2\Big]\end{aligned}\right\}\le C\Big(h^{2\alpha}+\frac{\sigma^2}{N}\Big),$$

i.e., the MSE of the FPMC estimator is asymptotically the same as the MSE of the standard MC estimator.

Proof. The bound for the standard MC estimator can be deduced from the bias-variance decomposition (Lemma 2.5) and assumption (2.1). The bound for the FPMC estimator is a corollary of Theorem 4.3. □

The corollary can be informally explained as follows: If assumption (4.3) holds then the MC estimate can be computed in finite precision arithmetic with a sufficiently small unit roundoff δ without losing the assymptotic accuracy.

A crucial question is when the assumptions (4.3) and (4.4) are satisfied. An answer to this question will be given in the following lemma.

Lemma 4.5. *Let f be sufficiently smooth so that Theorem 1.16 holds. Let $\widehat{u}_h$ be the solution of the discrete AVP with random data (Problem 1.13) computed effectively to precision δ; see* (4.1)*. Then*

$$\begin{aligned}|\mathbb{E}[Q_{h,\delta}-Q]| &= O(h^2+\delta) \quad \text{for } h,\delta\to 0,\\ var[Q_{h,\delta}] &= var[Q_h]+O(\delta) \quad \text{for } \delta\to 0;\end{aligned}$$

in other words, the assumptions (4.3) *and* (4.4) *from Theorem 4.4 hold.*

Proof. The bias error can be decomposed as follows:

$$\begin{aligned}|\mathbb{E}[Q_{h,\delta}-Q]| &\le \mathbb{E}[|G(\widehat{u}_h)-G(u)|]\\ &= \|G(u)-G(u_h)+G(u_h)-G(\widehat{u}_h)\|_{L^1(\Omega)}\\ &\le \|G(u)-G(u_h)\|_{L^1(\Omega)}+\|G(u_h)-G(\widehat{u}_h)\|_{L^1(\Omega)},\end{aligned}\tag{4.5}$$

where we used Jensen's inequality and the definition of the mean. The fact that

$$\|G(u) - G(u_h)\|_{L^1(\Omega)} \leq Ch^2, \tag{4.6}$$

where $C > 0$ is independent of h, u, and ω, follows from Theorem 1.16. Further, due to Lemma 1.19 and the fact that $\widehat{u}_h$ is computed effectively to precision δ, we get

$$\Big|G(u_h(\cdot,\omega)) - G(\widehat{u}_h(\cdot,\omega))\Big| \leq C\|f(\cdot,\omega)\|_{L^2(D)}\delta,$$

for a generic constant $C > 0$. Integrating this inequality over Ω yields

$$\|G(u_h) - G(\widehat{u}_h)\|_{L^1(\Omega)} \leq C\delta,$$

where C is independent of u, h, and ω. This, combined with (4.6) and using (4.5), gives us the desired estimate.

Let us now estimate the variance $\operatorname{var}[Q_{h,\delta}]$. We have

$$\begin{aligned}
\operatorname{var}[Q_{h,\delta}] &= \mathbb{E}\Big[(Q_{h,\delta} - \mathbb{E}[Q_{h,\delta}])^2\Big] = \|Q_{h,\delta} - \mathbb{E}[Q_{h,\delta}]\|^2_{L^2(\Omega)} \\
&\leq \Big(\|Q_h - \mathbb{E}[Q_h]\|_{L^2(\Omega)} + \|Q_{h,\delta} - Q_h\|_{L^2(\Omega)} + \Big|\mathbb{E}[Q_h - Q_{h,\delta}]\Big|\Big)^2 \\
&\leq \|Q_h - \mathbb{E}[Q_h]\|^2_{L^2(\Omega)} + 2\|Q_h - \mathbb{E}[Q_h]\|_{L^2(\Omega)}a + a^2 \\
&= \operatorname{var}[Q_h] + 2\sqrt{\operatorname{var}[Q_h]}a + a^2,
\end{aligned} \tag{4.7}$$

where

$$a \coloneqq \|Q_{h,\delta} - Q_h\|_{L^2(\Omega)} + \mathbb{E}\Big[|Q_h - Q_{h,\delta}|\Big].$$

In the same way as above, using Lemma 1.19 and the fact that $\widehat{u}_h$ is computed effectively to precision δ, it can be proved that $a \leq C\delta$ where C is independent of u, h, and ω. Using (4.7), we thus obtain

$$\begin{aligned}
\operatorname{var}[Q_{h,\delta}] &\leq \operatorname{var}[Q_h] + 2\sqrt{\operatorname{var}[Q_h]}c\delta + c^2\delta^2 \\
&= \operatorname{var}[Q_h] + \sqrt{\operatorname{var}[Q_h]}O(\delta).
\end{aligned}$$

The first possibility is to finish the proof here. In that case we would have $\operatorname{var}[Q_{h,\delta}] = \operatorname{var}[Q_h] + O(\delta)$ where the hidden constant depends on the variance $\operatorname{var}[Q_h]$. Since the dependence is known explicitly, we can directly use it in further analysis. We will, however, continue with the proof to make the constant independent of the variance. We prove that the variance $\operatorname{var}[Q_h]$ converges to $\operatorname{var}[Q]$ as $h \to 0$ and thus it can be bounded independently of h for h sufficiently small.

We know from Theorem 1.16 that

$$\mathbb{E}\Big[|Q_h - Q|^2\Big] \leq Ch^4. \tag{4.8}$$

Therefore $\mathbb{E}\Big[|Q_h - Q|^2\Big] \to 0$ as $h \to 0$. By using Jensen's inequality and Hölder's inequality we obtain that $\Big|\mathbb{E}[Q_h - Q]\Big| \to 0$ as $h \to 0$, which means that $\mathbb{E}[Q_h] \to \mathbb{E}[Q]$ as $h \to 0$. From the continuity of the L^2 norm we also have $\mathbb{E}[Q_h^2] \to \mathbb{E}[Q^2]$ as $h \to 0$. In all, we obtain that $\operatorname{var}[Q_h] = \mathbb{E}[Q_h^2] - \Big(\mathbb{E}[Q_h]\Big)^2$ converges to $\operatorname{var}[Q]$ as $h \to 0$. From this it follows that $\operatorname{var}[Q_h]$ can be, for h sufficiently small, bounded by a constant (uniformly with respect to h). From this and (4.8) it follows that $\operatorname{var}[Q_{h,\delta}] = \operatorname{var}[Q_h] + O(\delta)$ where the hidden constant is independent of h. □

In this lemma we verified the assumptions of Theorem 4.3 and Theorem 4.4 for our model problem (Problem 2.1). The error estimate for the finite precision MC estimator for our model problem (using the FEM solution) thus follows as a corollary of Theorem 4.3.

Corollary 4.6 (Error of the FPMC FE method)**.** *Let the assumptions of Lemma 4.5 be satisfied. Then for any $N \in \mathbb{N}$, $\delta > 0$, and $h > 0$, the error of the corresponding FPMC estimator $\widehat{Q}_{h,N,\delta}$ satisfies*

$$\mathbb{E}\Big[(\mathbb{E}[Q] - \widehat{Q}_{h,N,\delta})^2\Big] \leq C\left(\left(h^4 + \frac{\sigma^2}{N}\right) + \left(h^\alpha \delta + \delta^2 + \frac{\delta}{N}\right)\right).$$

Proof. The statement follows from Lemmas 4.5 and 4.3. □

4.2 Mixed precision MLMC method

The topic of this section will be the use of finite precision arithmetic in the MLMC method. Intuitively, the MLMC method offers better opportunities to exploit low precision arithmetic than the MC method, since a considerable part of the work in the MLMC method is done on the coarse levels where low precision can be efficiently used. We proceed to the definition of the mixed precision MLMC estimator.

Definition 4.7 (Mixed precision MLMC estimator)**.** *Let $\mathbb{E}[Q]$ be the quantity of interest defined in Problem 2.1. Let $h_0 \geq \ldots \geq h_L > 0$ be the discretisation parameters, $\delta_0, \ldots, \delta_L > 0$ a sequence of precisions, and $\widehat{u}_{h_l,\delta_l}$ the corresponding approximate solutions of the discrete random AVP (Problem 1.13) computed effectively to precision δ_l (see (4.1)). Let Q_{h_l,δ_l}, $l \in \{0, \ldots, L\}$, be the corresponding random variables defined analogously as in Definition 4.1. Define the following auxiliary MC estimators:*

$$\widehat{Y}_0 = \widehat{Y}_{h_0,N_0,\delta_0} \coloneqq \frac{1}{N_0} \sum_{k=1}^{N_l} Q^{(k)}_{h_0,\delta_0},$$
$$\widehat{Y}_l = \widehat{Y}_{h_l,N_l,\delta_l} \coloneqq \frac{1}{N_l} \sum_{k=1}^{N_l} \Big(Q^{(k)}_{h_l,\delta_l} - Q^{(k)}_{h_{l-1},\delta_{l-1}}\Big), \quad l = 1 \ldots, L.$$

Then, the estimator

$$\widehat{Q}^{MPML}_{L,\{N_l\},\{\delta_l\}} \coloneqq \sum_{l=0}^{L} \widehat{Y}_l$$

will be referred to as the mixed precision MLMC (MPMLMC) estimator for $E[Q]$.

4.2.1 Error estimates

The aim of this section is to analyse the error induced by using finite precision arithmetic in the MPMLMC method. The process will be similar to the error analysis of the standard MLMC method. We proceed, similarly to the standard MLMC method, to the bias-variance decomposition.

Lemma 4.8 (Bias-variance decomposition for the MPMLMC estimator)**.** *The mean square error of the MPMLMC estimator $\widehat{Q}^{MPML}_{L,\{N_l\},\{\delta_l\}}$ from Definition 4.7 can be expanded as*

$$\mathbb{E}\Big[(\mathbb{E}[Q] - \widehat{Q}^{MPML}_{L,\{N_l\},\{\delta_l\}})^2\Big] = (\mathbb{E}[Q - Q_{h_L,\delta_L}])^2 + \sum_{l=0}^{L} \frac{var[Y_l]}{N_l}.$$

Proof. The proof is identical to the proof of Lemma 2.9. □

We now proceed directly to the error estimate, which will help us to determine in which parts of the computation mixed precision can be efficiently used. It is again general in the sense that the discrete solution does not have to be the FE solution defined in Chapter 1, but any discrete solution satisfying certain assumptions.

Theorem 4.9 (Error of the MPMLMC method)**.** *Let $m \in \mathbb{N}$, $m > 1$, and let $h_0, h_1, \dots$ be discretization parameters satisfying $h_0 > 0$ and $h_l = \frac{1}{m} h_{l-1}$. Let $\delta_0, \delta_1, \dots$ be a sequence of precisions, i.e., $1 > \delta_l > 0$. Assume that there exist $\alpha, \beta, \gamma > 0$ such that $\alpha \geq \frac{1}{2}\min\{\beta, \gamma\}$ and*

$$|\mathbb{E}[Q_{h_l,\delta_l} - Q]| = O(h_l^\alpha + \delta_l), \tag{4.9}$$

$$var[Y_{h_l,N_l,\delta_l}] = O(h_l^\beta + \delta_l^2), \tag{4.10}$$

where the notation is the same as in Definition 4.7. Let $L \in \mathbb{N}$ and $N_0, \dots, N_L \in \mathbb{N}$ and let $\widehat{Q}^{MPML}_{L,\{N_l\},\{\delta_l\}}$ be the corresponding MPMLMC estimator. Then the MSE of this estimator satisfies

$$\mathbb{E}\Big[(\mathbb{E}[Q] - \widehat{Q}^{MPML}_{L,\{N_l\},\{\delta_l\}})^2\Big] \leq C\bigg(\Big(h_L^{2\alpha} + \sum_{l=0}^{L} \frac{h_l^\beta}{N_l}\Big) + \Big(h_L^\alpha \delta_L + \delta_L^2 + \sum_{l=0}^{L} \frac{\delta_l^2}{N_l}\Big)\bigg).$$

Proof. The inequality follows from a simple calculation: From the bias-variance decomposition (Lemma 4.8) and the assumptions (4.9) and (4.10) it follows that

$$\begin{aligned}
\mathbb{E}\Big[(\mathbb{E}[Q] - \widehat{Q}^{\mathrm{MPML}}_{L,\{N_l\},\{\delta_l\}})^2\Big] &= (\mathbb{E}[Q - Q_{h_L,\delta_L}])^2 + \sum_{l=0}^{L} \frac{\mathrm{var}[Y_l]}{N_l} \\
&\leq C\bigg((h_L^\alpha + \delta_L)^2 + \sum_{l=0}^{L} \frac{h_l^\beta + \delta_l^2}{N_l}\bigg) \\
&\leq C\bigg(\Big(h_L^{2\alpha} + \sum_{l=0}^{L} \frac{h_l^\beta}{N_l}\Big) + \Big(h_L^\alpha \delta_L + \delta_L^2 + \sum_{l=0}^{L} \frac{\delta_l^2}{N_l}\Big)\bigg),
\end{aligned}$$

where $C > 0$ is a generic constant. □

This theorem can be further simplified if we make further assumptions on the precision δ_l.

Corollary 4.10 (Error of the MPMLMC method 2)**.** *Let the MPMLMC estimator $\widehat{Q}^{MPML}_{L,\{N_l\},\{\delta_l\}}$ satisfy the assumptions of Theorem 4.9. Let $\widehat{Q}^{ML}_{L,\{N_l\}}$ be the standard MLMC estimator defined by the same values of h_l and N_l as the MPMLMC estimator. Assume that the MLMC estimator $\widehat{Q}^{ML}_{L,\{N_l\}}$ satisfies* (2.8) *and* (2.9)*, i.e.,*

we have the standard bias and variance decay. If we assume $\delta_L = O(h_L^\alpha)$ *and* $\delta_l^2 = O(h_l^\beta)$ *then the mean squared errors of both estimators can be bounded by*

$$\left.\begin{aligned}\mathbb{E}\Big[(\mathbb{E}[Q] - \widehat{Q}^{MPML}_{L,\{N_l\},\{\delta_l\}})^2\Big]\\ \mathbb{E}\Big[(\mathbb{E}[Q] - \widehat{Q}^{ML}_{L,\{N_l\}})^2\Big]\end{aligned}\right\} \leq C\Big(h_L^{2\alpha} + \sum_{l=0}^{L} \frac{h_l^\beta}{N_l}\Big),$$

i.e., the MSE of the MPMLMC estimator is asymptotically the same as the MSE of the standard MLMC estimator.

Proof. The bound for the standard MLMC estimator follows from the bias-variance decomposition (Lemma 2.9) and assumptions (2.8) and (2.9). The bound for the MPMLC estimator follows from the fact that $\delta_L = O(h_L^\alpha)$ and $\delta_l^2 = O(h_l^\beta)$ and Theorem 4.9. □

From Theorem 4.9, we see that the error can be naturally split into two parts. The first part bounds the error of the standard MLMC method and the second part comes from the fact that we use finite precision arithmetic. We observe that the error induced by using finite precision arithmetic can be further split: The first part, i.e., $h_L^\alpha \delta_L + \delta_L^2$, corresponds to the bias error and it tells us how much at most the bias error can be increased using finite precision arithmetic. We can see that this part of the error depends only on the precision δ_L which suggests that the bias error is influenced only by the precision used on the finest level of the computation.

Then there is the part of the error corresponding to the variance, i.e.,

$$\sum_{l=0}^{L} \frac{\delta_l^2}{N_l}.$$

We observe that the more samples we take on level l the lower precision δ_l can be used to achieve the same error on each level. Note that in this context "to lower the precision" means "to increase δ_l". This is particularly interesting in the case of our model problem when $\beta > \gamma$ and the lower precision can be exploited to reduce the computational time significantly as discussed in the following section.

4.2.2 Mixed precision MLMC FE method

This section aims to apply the general estimates derived above in the case of our model problem. Let us now verify that the assumptions (4.9) and (4.10) are satisfied in the case of our model problem.

Lemma 4.11. *Let* f *be sufficiently smooth so that Theorem 1.16 holds. Let* $m \in \mathbb{N}$, $m > 1$, *and let* $h_0, h_1, \ldots$ *be discretisation parameters satisfying* $h_0 > 0$ *and* $h_l = \frac{1}{m} h_{l-1}$. *Let* $\widehat{u}_{h_l}$ *be the solution of the discrete AVP with random data (Problem 1.13) computed effectively to precision* δ_l *(see* (4.1)*) and assume that there exists* $k_1, k_2 > 1$ *such that* $k_1\delta_l \leq \delta_{l-1} \leq k_2\delta_l$ *for all* $l \geq 1$. *Then*

$$|\mathbb{E}[Q_{h_l,\delta_l} - Q]| = O(h_l^2 + \delta_l), \tag{4.11}$$

$$var[Y_{h_l,N_l,\delta_l}] = O(h_l^4 + \delta_l^2). \tag{4.12}$$

Proof. The estimate (4.11) has already been verified in Lemma 4.5. Let us verify (4.12) (the idea is similar). Let us estimate

$$\begin{aligned}\operatorname{var}[Y_{h_l,N_l,\delta_l}] &= \mathbb{E}[Y^2_{h_l,N_l,\delta_l}] - \mathbb{E}[Y_{h_l,N_l,\delta_l}]^2 \\ &\leq \mathbb{E}\Big[\Big(Q_{h_l,\delta_l} - Q_{h_l} + Q_{h_l} - Q + Q - Q_{h_{l-1}} + Q_{h_{l-1}} - Q_{h_{l-1},\delta_{l-1}}\Big)^2\Big].\end{aligned} \tag{4.13}$$

Using the same technique as in the proof of Theorem 2.11, we obtain an estimate of the form

$$\begin{aligned}\operatorname{var}[Y_{h_l,N_l,\delta_l}] \leq & C\Big(\mathbb{E}[(Q_{h_l,\delta_l} - Q_{h_l})^2] + \mathbb{E}[(Q_{h_l} - Q)^2] \\ & + \mathbb{E}[(Q - Q_{h_{l-1}})^2] + \mathbb{E}[(Q_{h_{l-1}} - Q_{h_{l-1},\delta_{l-1}})^2]\Big).\end{aligned} \tag{4.14}$$

The quantity $\mathbb{E}[(Q_{h_l} - Q)^2] + \mathbb{E}[(Q - Q_{h_{l-1}})^2]$ can be estimated analogously as in the proof of Theorem 2.11 by

$$\mathbb{E}[(Q_{h_l} - Q)^2] + \mathbb{E}[(Q - Q_{h_{l-1}})^2] \leq C h_l^4. \tag{4.15}$$

The quantity $\mathbb{E}[(Q_{h_l,\delta_l} - Q_{h_l})^2] = \|G(u_{h_l}) - G(\widehat{u}_{h_l})\|^2_{L^2(\Omega)}$ can be estimated from above as follows: Due to Lemma 1.19 and the fact that $\widehat{u}_{h_l}$ is computed effectively to precision δ_l, we get

$$\Big|G(u_{h_l}(\cdot,\omega)) - G(\widehat{u}_{h_l}(\cdot,\omega))\Big| \leq C\|f(\cdot,\omega)\|_{L^2(D)}\delta_l,$$

for a generic constant $C > 0$. Taking the second power of this inequality and integrating over Ω yields

$$\|G(u_{h_l}) - G(\widehat{u}_{h_l})\|^2_{L^2(\Omega)} \leq C\delta_l^2, \tag{4.16}$$

where C is independent of u, h_l, and ω. Using (4.16) we obtain

$$\begin{aligned}\|G(u_{h_l}) - G(\widehat{u}_{h_l})\|^2_{L^2(\Omega)} + \|G(u_{h_{l-1}}) - G(\widehat{u}_{h_{l-1}})\|^2_{L^2(\Omega)} &\leq C(\delta_l^2 + \delta_{l-1}^2) \\ &\leq C(\delta_l^2 + k_2^2\delta_l^2) \\ &\leq C\delta_l^2.\end{aligned}$$

This, together with (4.15) and (4.13), yields the desired estimate (4.12). □

This lemma allows us to reformulate Theorem 4.9 specifically for the discrete solution obtained by the finite element method, which is summarized in the following corollary.

Corollary 4.12 (Error of the MPMLMC FEM)**.** *Let the assumptions of Lemma 4.11 be satisfied. Let $L \in \mathbb{N}$ and $N_0, \dots, N_L \in \mathbb{N}$ and let $\widehat{Q}^{MPML}_{L,\{N_l\},\{\delta_l\}}$ be the corresponding MPMLMC estimator. Then the MSE of this estimator satisfies*

$$\mathbb{E}\Big[(\mathbb{E}[Q] - \widehat{Q}^{MPML}_{L,\{N_l\},\{\delta_l\}})^2\Big] \leq C\bigg(\Big(h_L^4 + \sum_{l=0}^{L}\frac{h_l^4}{N_l}\Big) + \Big(h_L^2\delta_L + \delta_L^2 + \sum_{l=0}^{L}\frac{\delta_l^2}{N_l}\Big)\bigg).$$

Proof. The claim follows from Lemma 4.11 and Theorem 4.9. □

Note that in this section we have not focused on the cost of the MPMLMC method. This is because in terms of FLOPs the asymptotic cost remains the same as for the standard MLMC method as long as $\delta_L = O(h_L^\alpha)$ and $\delta_l^2 = O(h_l^\beta)$. However, due to the use of lower precision arithmetic, the overall computational time can be reduced significantly which is the topic of the next section.

4.2.3 Cost analysis

The goal of this section is to estimate the gain from using the MPMLMC method in terms of the computational time. Note that this section does not aim to be exact, but rather to estimate realistically the resulting speedup. The verification of these results in practice using real low precision hardware is outside the scope of this thesis.

To obtain the reference values we use the standard MLMC method where all computations are carried out in double precision. We exploit the fact that in the setting when $\beta > \gamma$, which is true for the MLMC FE method (see Theorem 2.11), the variance decays faster than the cost increases and thus the cost on the coarsest level dominates. Due to the results from Section 4.2.1 the linear system on the coarsest level, computed in total N_0 times, can be solved very "inaccurately". As a result the overall computational time is dramatically reduced. Let us now formulate these ideas more precisely.

If $\beta > \gamma$ then we know from Section 2.3.3 that the cost on the coarsest level dominates and the cost per level decays with the factor $m^{\frac{\gamma-\beta}{2}}$ where $m = h_{l-1}/h_l$. Recall that in the case of MLMC FEM we have $\beta = 4$ and $\gamma = 2$ (see the proof of Theorem 2.11) so that $m^{\frac{\gamma-\beta}{2}} = m^{-1}$. This suggests that the overall cost (in terms of FLOPs) on the finest level L is small compared to the cost on levels $l = 0, \dots, L-1$ and we neglect it in this section. At the same time Theorem 4.10 tells us that the system of linear equations coming from the FE method (the FE system) can be solved effectively to precision δ_l satisfying $\delta_l^2 = O(h_l^\beta)$ and the asymptotic error of the MPMLMC estimator remains the same as when the linear system is solved exactly. Moreover, (3.8) and the discussion in Section 3.4 suggest that if an algorithm is run in quarter, half, or single precision, the overall computational time can be reduced by the factor 8, 4, or 2, respectively, compared to the same algorithm in double precision. The combination of these observations yields the following corollary.

Corollary 4.13. *Let $\beta > \gamma$ and let the assumptions of Theorem 4.10 be satisfied. Assume moreover that there exists a precision δ satisfying $\delta^2 = \delta_l^2 = O(h_l^\beta)$ for all $l = 0, \dots, L-1$. Suppose that δ is such that the FE system can be solved on each level $l = 0, \dots, L-1$ effectively to precision δ using quarter (or half or single) precision for the dominant part of the computations and let $\widehat{Q}^{MPML}_{L,\{N_l\},\{\delta_l\}}$ be the corresponding MPMLMC estimator. Assume further that the reference standard MLMC estimator is computed using double precision on all levels.*

Then the MSE of the MLMC and MPMLMC estimator is asymptotically the same in the sense of Theorem 4.10 and the computational time is reduced by approximately $8\times$ (or $4\times$ or $2\times$) using the MPMLMC estimator. In other words,

$$T(\widehat{Q}^{MPML}_{L,\{N_l\},\{\delta_l\}}) \approx \frac{1}{C} T(\widehat{Q}^{ML}_{L,\{N_l\}}),$$

where $T(\cdot)$ denotes the computational time and $C = 8$ (or $C = 4$ or $C = 2$, depending on what precision is used for the dominant part of the computations).

Let us further explain the assumption that the linear system should be solved effectively to precision δ using a certain precision for the dominant part of the computations. To achieve this, iterative refinement can be efficiently used. As

discussed in Section 3.3, iterative refinement allows us to use a lower precision for the costly, dominant part of solving the equation (e.g., for the factorization) and then regain the desired accuracy using relatively cheap iterations.

Naturally, the MPMLMC method can be also used in a setting when $\beta \leq \gamma$. However, we do not expect the computational time to be reduced as dramatically as in the case $\beta > \gamma$. In general we can say that if the dominant part of solving the FE system on the finest level is carried out at least in single precision, the computational time is in the same way reduced at least $2\times$ compared to double precision.

4.2.4 Adaptive MPMLMC algorithm

In this section we develop an adaptive algorithm which will automatically choose the correct precision on each level of the MLMC method. We will use Algorithm 1 as the foundation for our proposed algorithm. The first step is the choice of the precision.

In order to choose the correct precision in each step, we will use the error bound for the MPMLMC method from Theorem 4.9. We propose the following choice: Choose the precision δ_l on level l such that the total MSE of the MPMLMC estimator is not greater than a constant times the MSE of the MLMC estimator for a fixed constant from the interval $(1, 2)$. According to Theorem 4.9, for this to hold it suffices to choose $\delta_l,\ l = 0, \dots, L$ such that

$$h_L^\alpha \delta_L + \delta_L^2 + \sum_{l=0}^{L} \frac{\delta_l^2}{N_l} \leq k_p \Big(h_L^{2\alpha} + \sum_{l=0}^{L} \frac{h_l^\beta}{N_l} \Big)$$

for a fixed constant $k_p \in (0, 1)$. To balance the terms in the error estimate, it is sufficient to choose δ_L such that

$$h_L^\alpha \delta_L + \delta_L^2 \leq \frac{1}{2} k_p h_L^{2\alpha} \tag{4.17}$$

and $\delta_l,\ l = 0, \dots, L-1$ such that

$$\sum_{l=0}^{L} \frac{\delta_l^2}{N_l} \leq \frac{1}{2} k_p \sum_{l=0}^{L} \frac{h_l^\beta}{N_l}. \tag{4.18}$$

Since for $\delta_L \leq h_L^\alpha$ is $\delta_L^2 \ll h_L^\alpha \delta_L$, it suffices to choose

$$\delta_L \leq \frac{1}{2} k_p h_L^\alpha.$$

Moreover, in order to satisfy (4.18) we can choose δ_l as

$$\begin{aligned} \delta_l &\coloneqq \sqrt{\frac{k_p}{2}} h_l^{\beta/2}, \quad l = 0, \dots, L-1, \\ \delta_L &\coloneqq \min\left\{ \sqrt{\frac{k_p}{2}} h_L^{\frac{\beta}{2}}, \frac{k_p}{2} h_L^\alpha \right\}. \end{aligned} \tag{4.19}$$

By this choice both (4.17) and (4.18) are satisfied and we obtain the desired error estimate if the rest of the assumptions in Theorem 4.9 are satisfied. The precise formulation is given for the discrete solution obtained by the finite element method in the following example.

Example 4.14. Let f be sufficiently smooth so that Theorem 1.16 holds. Let $m \in \mathbb{N}$, $m > 1$, and let $h_0, h_1, \ldots$ be discretisation parameters satisfying $h_0 > 0$ and $h_l = \frac{1}{m} h_{l-1}$. Let $\widehat{u}_{h_l}$ be the solution of the discrete AVP with random data (Problem 1.11) computed effectively to precision δ_l where δ_l is given by (4.19) using a fixed $k_p \in (0, 1)$. Let $L \in \mathbb{N}$ and $N_0, \ldots, N_L \in \mathbb{N}$ and let $\widehat{Q}^{\mathrm{MPML}}_{L,\{N_l\},\{\delta_l\}}$ be the corresponding MPMLMC estimator. Then the MSE of this estimator satisfies

$$\mathbb{E}\Big[(\mathbb{E}[Q] - \widehat{Q}^{\mathrm{MPML}}_{L,\{N_l\},\{\delta_l\}})^2\Big] \leq C(1 + k_p)\Big(h_L^4 + \sum_{l=0}^{L} \frac{h_l^4}{N_l}\Big). \tag{4.20}$$

The proof of this claim follows from the definition of δ_l (see (4.19)) and Corollary 4.12.

Let us discuss in more detail the choice of the constant k_p. Although in this work the constant k_p is chosen to be fixed, more general choices are possible, one of which is discussed in the next paragraph. Note that the precision δ_l defined by (4.19) is, for $l < L$, not sensitive with respect to the changes of the constant k_p. Indeed, if the constant is decreased $100\times$, the precision decreases only $10\times$. Based on the bound (4.20), the value $k_p \coloneqq 1/10$ is a safe choice to bound the mixed precision error, as demonstrated in Chapter 5. Note also that the values of δ_l can be computed "on the fly" with no additional cost.

It is natural to ask how the choice of the constant k_p affects the number of samples N_l required on each level l to achieve the desired tolerance. According to Theorem 4.9 and the discussion below, if the precision δ_l is chosen according to (4.19) then the variance is increased on each level at most by approximately the factor $(1 + k_p)$. This means, according to (2.22), that the number of samples N_l on each level is increased at most by the same factor $(1 + k_p)$. Since $k_p \ll 1$, this does not pose a problem for us. In some cases however, it might useful to control how the number of samples N_l increases. In such a case we can choose the constant as $k_p \propto 1/N_l$. This will ensure using again Theorem 4.9 and (2.22) that the number of samples N_l increases on each level at most by an additive constant. We proceed now to the formulation of the MPMLMC algorithm, given in Algorithm 4. See Section 2.3.2 for the notation.

Throughout this chapter we have assumed that we are able to compute the approximate discrete solution $\widehat{u}_{h_l}$ on the level l effectively to precision δ_l, i.e., we have assumed (4.1). Now we address the question how to achieve this. We propose two ways which can possibly be combined: First, any suitable iterative solver can in principle be used with the stopping criterion given by (4.2). Since the values of δ_l obtained using the adaptive MPMLMC algorithm are typically relatively "big" (see Chapter 5 for examples), the iterative solver can potentially achieve the tolerance in a very small number of iterations, leading to significant gains in computational time. Note that this case is not covered by the cost analysis in the previous section and in Corollary 4.13. In this case, the gain in computational time does not come primarily from using low precision, but rather from reducing the number of iterations.

The second possibility how to achieve the desired tolerance is the use of iterative refinement and this will be our choice in Chapter 5. Iterative refinement can be used with in principle with any direct or iterative solver; see Section 3.3 and [5]. We use it with the $LDL^\top$ solver. We illustrate the utility of iterative refinement in our context by an example.

Algorithm 4: Adaptive MPMLMC algorithm

Input: h_0, m, ϵ, $L = 1$, $L_{\max}$, $N_0 = N_1 = N_{\text{init}}$

Output: $\widehat{Q}^{\text{MPML}}_{L,\{N_l\},\{\delta_l\}}$

while $L \leq L_{max}$ **do**

  Compute δ_l, $l = 0, \ldots, L$, using (4.19);

  for $l = 0$ **to** L **do**

    Compute N_l new samples $Y_l^{(k)}$ using Def. 4.7;

    Compute $\widehat{Y}_l$, s_l^2 and estimate C_l;

  end

  Update estimates for N_l using (2.22);

  if $|\widehat{Y}_L| > \frac{rm^{\alpha}-1}{\sqrt{2}}\epsilon$ **then**

    $L \coloneqq L + 1$;

    $N_L \coloneqq N_{\text{init}}$;

  if $|\widehat{Y}_L| \leq \frac{rm^{\alpha}-1}{\sqrt{2}}\epsilon$ *and* $\sum_{l=0}^{L} s_l^2/N_l \leq \epsilon^2/2$ **then**

    $\widehat{Q}^{\text{MPML}}_{L,\{N_l\},\{\delta_l\}} \coloneqq \sum_{l=0}^{L} \widehat{Y}_l$;

end

Suppose that we are in the setting of our model problem (Problem 2.1) and assume that the discrete solution corresponds to the solution of the discrete AVP with random data (Problem 1.11). This means, according to Lemma 4.11, that the assumptions of Theorem 4.12 are satisfied, namely, $\beta = 4$. Let $h_l = 1/32$ where $l < L$. Then we obtain from (4.19), assuming $k_p = 0.1$, that $\delta_l \approx 2.2 \times 10^{-4}$. In order to obtain the approximate discrete solution $\widehat{u}_{h_l}$, we have to now solve a linear system, let us denote it $Ax = b$. For the solution $\widehat{u}_{h_l}$ to be computed effectively to precision δ_l, we have to assure that the relative residual of the system is less than $\delta_l \approx 2.2 \times 10^{-4}$. Suppose that the system is solved using a direct solver. Because $\delta_h \approx 5 \times 10^{-4} > \delta_l$, we cannot expect that it will be sufficient to use the direct solver in half precision to achieve the desired accuracy. Instead, we have to move directly to single precision with the unit roundoff $\delta_s \approx 6 \times 10^{-8}$ which typically assures that the system is solved accurately enough. However, this approach does not allow us to use the full potential of the MPMLMC method, since the accuracy of the computed solution is unnecessarily high.

The previous example illustrates the problem coming from the "sparsity" of the standard precision formats - quarter, half, single, and double. To tackle this problem we use iterative refinement, described in Section 3.3. As explained in the aforementioned section and as illustrated by numerous examples in Chapter 5, iterative refinement helps us to achieve just the desired accuracy with a small additional cost. If we come back to the previous example, it turns out that in order to achieve the desired accuracy $\delta_l \approx 2.2 \times 10^{-4}$ it is sufficient to factorize the matrix A in half precision and then perform one iteration of iterative refinement where only the residual is computed in single precision (see Algorithm 3). It is now clear that the cost of using iterative refinement in this case stays approximately the same as the cost of running just the sparse direct solver once in half precision,

due to the fact that the factorization part dominates the overall cost; see Section 3.2.1.

5. Numerical results

In this chapter, we present numerical results to validate the theoretical findings we have discussed in the previous chapter on the mixed precision multilevel Monte Carlo method. To demonstrate our results, we will use two problems, one simple, where an analytic solution is available, allowing us to perform a comprehensive analysis of our method. The second problem will be a more realistic example, demonstrating the practical application of our approach. We begin by providing an overview of our implementation, detailing the technical aspects of our computations, including the hardware and software used.

We conduct all our numerical experiments using MATLAB R2019b on a computer equipped with an Intel(R) Core(TM) i5-3320M CPU and 8GB RAM. All the algorithms presented in this chapter are implemented by us except for the simulation of low precision arithmetic, for which we use the "chop" function written by Nicholas Higham [18] with the setting "q43" to obtain quarter precision. Apart from the chop function, the implementation of our algorithms is carried out using standard numerical libraries available in MATLAB.

5.1 Poisson's equation with random right-hand side

Let us formulate the first problem, which is a special case of Problem 1.10:

$$\begin{aligned} -\Delta u(\cdot,\omega) &= f(\cdot,\omega) \quad \text{on } D, \\ u(\cdot,\omega) &= 0 \quad \text{on } \partial D, \end{aligned} \tag{5.1}$$

where we take $D = (0,1)^2$. The weak formulation (see Problem 1.11) is given by

$$\tilde{a}\Big(u(\cdot,\omega), v\Big) = l(v,\omega) \quad \forall v \in V = H_0^1(D),$$

where

$$\begin{aligned} \tilde{a}\Big(u(\cdot,\omega), v\Big) &\coloneqq \int_D \nabla u(x,\omega)\cdot\nabla v(x)\mathrm{d}x \quad \text{and} \\ l(v,\omega) &\coloneqq \int_D f(x,\omega)v(x)\mathrm{d}x. \end{aligned}$$

We will be concered with the quantity of interest given by

$$Q = \int_\Omega \Big(\int_D u(x_1,x_2,\omega)\mathrm{d}(x_1,x_2)\Big)\mathrm{d}\omega.$$

In other words, the quantity of interest is the expected value of the random variable $\omega \mapsto G(u(\cdot,\omega))$ where the functional G is given for a fixed $\omega \in \Omega$ by

$$G(u(\cdot,\omega)) = \int_D u(x_1,x_2,\omega)\mathrm{d}(x_1,x_2).$$

In particular, we consider $f = -2\exp(c\omega)(x_1(x_1-1)+x_2(x_2-1))$ where $c > 0$ is a fixed constant and $\omega \sim \mathrm{Uni}(0,1)$. In this case, the solution is given

	$l=0$		$l=1$	
Prec. level	itref	δ_0	itref	δ_1
1	$qhhh0$	2.1e−1	$qhhh0$	6.2e−1
2	$qhhh0$	2.1e−1	$qhhh1$	3.8e−2
3	$qhhh1$	1.7e−2	$qhhh2$	6.5e−3
4	$qhhh2$	2.2e−3	$qhss3$	7.0e−4
5	$qhss3$	2.5e−4	$qhss4$	1.1e−4
6	$qhss4$	3.0e−5	$hsss1$	4.3e−6
7	$qhss6$	1.3e−6	$hsss2$	4.8e−7

Table 5.1: First example: MPMLMC settings and choice of the precisions for iterative refinement (Algorithm 3). The exact setting is described symbolically, e.g., $(\delta_f, \delta_7, \delta, \delta_r, n_{\text{it}}) = (hsss2)$, where δ_7 is the precision chosen at step 7 of Algorithm 3 and h = half, s = single. For example, line 7 in the table says that in test number 7 we used the "qhss" iterative refinement with 6 iterations on the coarser level $l = 0$ and we achieved accuracy 1.3e−6 in terms of the relative residual.

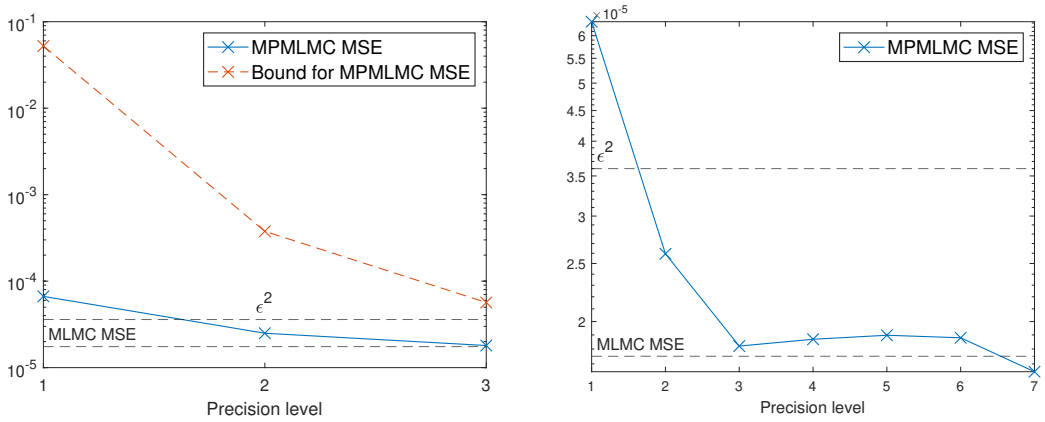
by $u = x_1(1-x_1)x_2(1-x_2)\exp(c\omega)$ and $Q = \frac{1}{36c}(\exp(c)-1)$. Throughout this section, we consider $c = 1.5$.

To obtain the discrete solution of (5.1) (for a fixed value of ω), we use the finite element method as described in Chapter 1, which includes solving the corresponding discrete abstract variational problem (Problem 1.13). To this end, we solve the resulting linear system (see Section 1.2.3) using methods from Chapter 3; the exact setting is always described when the numerical results are shown. To obtain the reference solution (in Algorithm 1), the built-in backslash function is used. Throughout our experiments the initial discretization parameter is $h_0 = 1/4$ and the mesh is refined on each level by a factor $m = 2$.

Note that in this case the assumptions 1.12 and 1.14 are trivially satisfied. Moreover, $f \in C^\infty(D)$ so that the assumptions of Theorem 1.16 and Lemma 1.19 are satisfied due to Example 1.17.

The first example of this section exploits the fact that the exact value of the quantity of interest is known, which enables a thorough analysis of the error of the MPMLMC estimator and comparison of its results to the standard MLMC estimator. To this end, let us choose the tolerance for the MLMC algorithm to be $\epsilon = 6 \times 10^{-3}$. The assumptions of theorem 2.10 are satisfied with $\alpha = 2$, $\beta = 4$ and $\gamma = 2$ as proved in Theorem 2.11. We choose the value $\gamma = 2$, since for the size of matrices which are of interest, the MATLAB backslash operator exhibits a linear convergence rate; see Figure 5.2. We use the same value of the parameter γ in the MPMLMC algorithm (Algorithm 4). This corresponds to the case when we use a similar solver to MATLAB backslash implemented in lower precision. Since MATLAB currently does not support half and quarter precision, we rely on our implementations described in Chapter 3. Further, we set the initial number of samples to $N_{init} = 100$.

In the setting described above, Algorithm 1 converges to the number of samples $N_0 = 50$ and $N_1 = N_L = 3$. We now aim to verify Theorem 4.9. Let us consider (in the whole example) the MPMLMC estimator $\widehat{Q}^{\text{MPML}}_{L,\{N_l\},\{\delta_l\}}$ corresponding to the values $N_0 = 50$ and $N_1 = N_L = 3$. To verify Theorem 4.9 we vary



(a) Precision level 1-3 (b) Precision level 1-7

Figure 5.1: Comparison of MLMC in mixed precision and MLMC in reference precision (double) on Poisson's problem with random RHS. Higher precision level means higher precision is used. The specific values of precision are given in Table 5.1. The MSE of the MPMLMC estimator decreases rapidly with increasing precision and soon (from level number 3 on) we are not able to distinguish between the finite precision and the probabilistic error.

the precisions δ_l and compare the observed error and the error bound given by the theorem. Recall that we need to ensure that the FE solution is computed effectively to precision δ_l in the sense of (4.1). Various values of precision δ_l are achieved by varying the settings of the iterative refinement algorithm (Algorithm 3). For the factorization method in step 1 of the algorithm we use the $LDL^\top$ decomposition so that we can avoid computing the square root on the diagonal in Algorithm 2. Note that in the case where only the right-hand side is random, the matrix decomposition can be precomputed. We use this fact to speed-up our $LDL^\top$ solver.

Let us describe now the exact setting of the iterative refinement algorithm. The algorithm contains 3 precisions, i.e., δ_f, δ, and δ_r that take on one of the values quarter (q), half (h), single (s), or double (d). The desired accuracy δ_l is achieved by prescribing a fixed number of iterations of iterative refinement. To simplify the notation, we describe the exact setting of the iterative refinement schematically as an ordered quintuple, e.g., $(\delta_f, \delta_7, \delta, \delta_r, n_{\text{it}}) = (hhss2)$, where δ_7 is the precision chosen at step 7 of the algorithm 3. The specific settings of iterative refinement used in this subsection on both level h_0 and level $h_1 = h_L$ along with the achieved accuracies δ_0 and $\delta_1 = \delta_L$ are shown in Table 5.1. The presented values of δ_l are computed as a sample average of 1000 samples using (4.2). Note that in this test we do not aim to use iterative refinement and low precision arithmetic to reduce the cost of the computations but merely as a tool to achieve the desired accuracy of computations.

The fact that we are able to determine the exact value of the quantity of interest allows us to estimate the mean squared error of both the MLMC and MPMLMC estimators. At first, we perform 3 experiments, starting at quarter precision and gradually increasing the precisions δ_l, and we analyse the results,

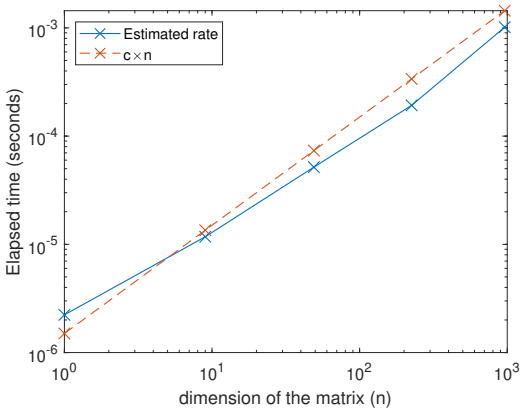


Figure 5.2: MATLAB backslash rate. We observe the linear rate for small matrices.

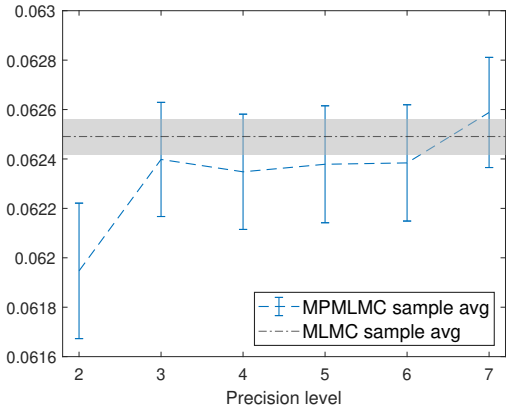


Figure 5.3: MPMLMC and MLMC sample average comparison. Higher precision level means higher precision is used. The specific values of precision are given in Table 5.1. From level number 3 onward we are not able to distinguish between the finite precision and probabilistic error.

which can be seen in Figure 5.1a. The solid line shows the MSE of the MPMLMC estimator using increasing levels of precision (using various settings of iterative refinement). The settings corresponding to the levels of precision labeled 1, 2, and 3, are shown in the first three rows of Table 5.1, respectively. As the reference value, the plot shows the MSE of the standard MLMC estimator. All of these four values of the MSE were computed using 10^4 samples of the corresponding estimator. The plot also shows the tolerance ϵ^2 and the bound given by Theorem 4.9. For the purpose of the plot we set the constant $C = 0.1$. The bound is, however, only asymptotic and therefore it is too rough in this preasymptotic phase.

We observe that the MPMLMC estimator performs unexpectedly well even when both of the equations are solved in quarter precision with no iterative refinement (precision level 1). As δ_l decreases, the MSE of the MPMLMC estimator approaches the MSE of the MLMC estimator as expected. Note that the MSE of the MPMLMC estimator achieves the desired tolerance already in precision level 2.

Let us investigate what happens when we decrease δ_l further. The MSE of the MPMLMC estimator is shown on Figure 5.1b along with the tolerance ϵ^2 and the MSE of the reference MLMC estimator. Each of the 7 results was obtained using 10^3 samples and the settings are given in Table 5.1, the reference value was computed using 10^4 samples. This plot suggests that starting from precision level 3, when δ_l decreases more, the probabilistic part of the error starts to dominate and we are no longer able to see the impact of finite precision arithmetic on the overall error. This is confirmed by Figure 5.3. The number of samples used remains the same but this time we show the sample average along with the asymptotic 95% confidence intervals instead of the MSE. We observe that starting from precision level 3, the confidence intervals of the MPMLMC estimator contain the whole

	$k_p = 0.1$			
ϵ	δ_0	δ_1	δ_2	δ_3
6e−3	1.4e−2	7.8e−4	-	-
3e−3	1.4e−2	3.5e−3	1.9e−4	-
1.5e−3	1.4e−2	3.5e−3	1.9e−4	-
7.5e−4	1.4e−2	3.5e−3	8.7e−4	4.9e−5

l	itref
0	$qhhh$
1	$hhss$
2	$hhss$
3	$hsss$

Table 5.2: Second example, $k = 0.1$: MPMLMC settings and choice of the precisions for iterative refinement (Algorithm 3). Note that the factorization is carried out in quarter precision on the coarsest level and in half precision on the other levels. Values $\delta_0, \ldots, \delta_3$ indicate the required effective precision in terms of relative residual norm on each level given by the adaptive algorithm (Algorithm 4).

confidence interval of the MLMC estimator so that no additional information regarding the impact of finite precision arithmetic can be obtained. Note that the length of the confidence interval is proportional to the reciprocal of the square root of the number of samples taken. Therefore making our error estimates more accurate increases the cost of computations significantly which makes the estimates too expensive to compute even though the matrix decomposition can be precomputed.

The second example of this section aims to verify Algorithm 4. To this end we use the same equation as in the previous test. The algorithm is tested using various values of the tolerance ϵ and of the parameter k_p and the results are shown in Figure 5.4. For each value of the tolerance ϵ, the MPMLMC estimator is computed using Algorithm 4 and the MSE is displayed (denoted MPMLMC MSE). The MSE is estimated using 100 samples for each estimator. For each of the MPMLMC estimators, the reference value of its MSE is computed using the MATLAB backslash function and the results are displayed (denoted MLMC MSE). These reference values are computed using 500 samples for each estimator. Tables 5.2 and 5.3 show the exact setting of the MPMLMC algorithm for test, i.e., the setting of the iterative refinement, k_p, and the required effective precision in terms of relative residual norm on each level, i.e., δ_l. Note that now the values δ_l are computed by the adaptive algorithm (Algorithm 4) and are used in the stopping criterion of the iterative refinement (Algorithm 3).

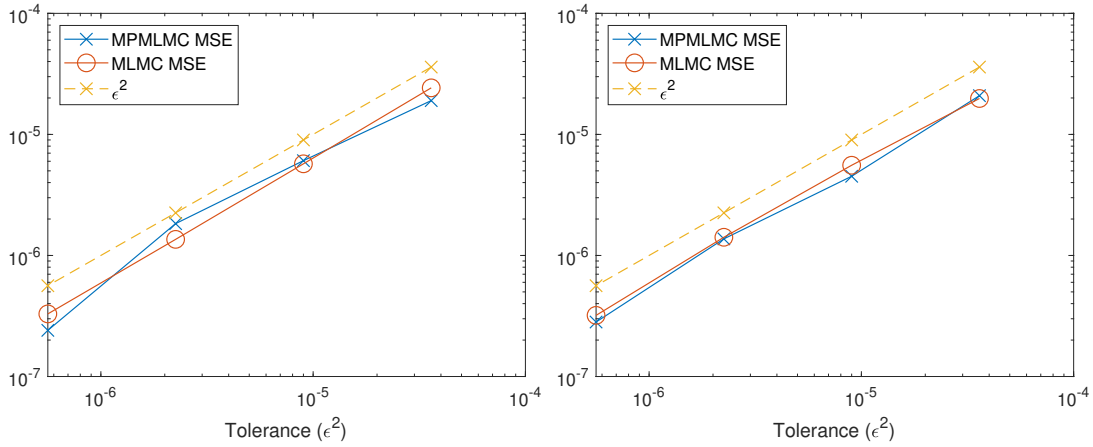
In this case we do not aim to analyse the error thoroughly as above. The goal is to compare the error with the tolerance and estimate the overall cost. The algorithm always reaches the desired tolerance, but a comparison of the results is difficult, since the probabilistic error plays a big part here. We observe that the MSE of the MPMLMC estimator is close to the MSE of the reference MLMC estimator. For some values we even observe that the MSE of the MPMLMC estimator is smaller than the MSE of the MLMC estimator. From our point of view the main cause is that the number of samples is not big enough and thus the probabilistic part of the error dominates, which was observed in the first example as well.

Let us now estimate the gain in terms of the cost. In Tables 5.2 and 5.3 we can see that the factorization is performed in quarter precision (level $l = 0$) or half precision (levels $l = 1, 2, 3$), which would give us, by Corollary 4.13, a speedup of $4\times$ to $8\times$ compared to the same algorithm run in double precision. However,

	$k_p = 0.4$			
ϵ	δ_0	δ_1	δ_2	δ_3
6e−3	2.8e−2	3.1e−3	-	-
3e−3	2.8e−2	7.0e−3	7.8e−4	-
1.5e−3	2.8e−2	7.0e−3	7.8e−4	-
7.5e−4	2.8e−2	7.0e−3	1.7e−3	2.0e−4

l	itref
0	$qhhh$
1	$hhss$
2	$hhss$
3	$hsss$

Table 5.3: Second example, $k = 0.4$: MPMLMC settings and choice of the precisions for iterative refinement.



(a) $k_p = 0.1$ (b) $k_p = 0.4$

Figure 5.4: Second example: MPMLMC MSE compared to the reference MLMC MSE (double precision) and the tolerance — Poisson's problem with random RHS.

since we can precompute the matrix factorization, the dominant computational cost is moved to the substitution part. In this case the same corollary gives us a 4× speedup, since the substitution is performed in half precision on all levels apart from the finest one, i.e., $l = 3$, and the cost on the finest level is negligible compared to the other levels. The estimated speedup is the same for both $k_p = 0.1$ and $k_p = 0.4$. The difference between them is that the former needs at some levels (not all) at most one additional iteration of the iterative refinement, which does not increase the cost significantly. A more significant gain in the cost can be obtained when the matrix factorization dominates, which will be the case in the following section. We note here that there is indeed a difference between taking $k_p = 0.1$ and $k_p = 0.4$ in terms of the number of iterations of the iterative refinement. For example, in the former case we need on the coarsest level $l = 0$ on average two iterations, whereas in the latter case only one iteration on average is required (see Table 5.1).

Let us make a final remark on the computational cost. While we recognise the need to verify our theoretical results in more detail, e.g., using smaller values of the tolerance ϵ or using more samples in Figures 5.4a and 5.4b, we are limited by the performance of our hardware (described above). To give an actual CPU time, the MSE of the MPMLMC estimator in Figure 5.4b was obtained in approximately 7.5 hours. Therefore, increasing the number of samples, e.g., by a

factor of 10, to lower the probabilistic error, is not within our reach.

5.2 Elliptic PDE with lognormal random coefficient

5.2.1 Introduction and motivation

One example of how the MLMC method is employed is modeling the flow through porous media in geosciences. An overview of this topic can be found in [25]. The simplest mathematical model for the flow is Darcy's law, which reads (technical details omitted):

$$\begin{aligned} q &= \frac{-k}{\mu}\nabla p \quad \text{on } D, \\ p &= p_0 \quad \text{on } \partial D. \end{aligned} \tag{5.2}$$

In this equation, the vector q refers to the volumetric flux or Darcy velocity, while the tensor k represents the permeability, a material parameter describing how easily water flows through the medium. The quantity μ, on the other hand, is the dynamic viscosity of the fluid, and p is the pressure of the fluid; both are scalar. For simplicity, only Dirichlet boundary conditions are considered, although Neumann boundary conditions are common as well; see [26]. Another condition on q arises from the law of mass conservation, i.e.,

$$\nabla \cdot q = 0. \tag{5.3}$$

Combining (5.2) and (5.3), we obtain Darcy's law in its primal form, i.e.,

$$\begin{aligned} -\nabla \cdot \left(\frac{k}{\mu}\nabla p\right) &= 0 \quad \text{on } D, \\ p &= p_0 \quad \text{on } \partial D. \end{aligned} \tag{5.4}$$

Note that this form of the equation is similar to our model example (Problem 1.1). The uncertainty in this equation arises from the parameter k representing the permeability, which has to be obtained empirically in practice and is measured only in a few locations and then extrapolated to the entire domain. Detailed information can be found in [25], [22], and [11].

We will make the common assumption that k is a lognormal random field (see Definition 1.9) and can be expanded using the Karhunen-Loève (KL) expansion. The KL expansion can be truncated so that the coefficient k is described using a finite-dimensional random vector. For more detailed information about the lognormal random fields, the KL expansion, and an example of their use, see [2], [23] and the references therein.

5.2.2 Problem statement

With this motivation in mind, we will now solve an equation of the following form, which is a special case of (1.9):

$$\begin{aligned} -\nabla \cdot \Big(a(\cdot,\omega)\nabla u(\cdot,\omega)\Big) &= f \quad \text{on } D, \\ u(\cdot,\omega) &= 0 \quad \text{on } \partial D, \end{aligned} \tag{5.5}$$

for a given random field a. The random field is chosen in such a way that it represents the truncated Karhunen-Loève expansion, i.e.,

$$a(x_1,x_2,\omega) = \exp\left(\sum_{j=1}^{s} \omega_j \frac{1}{j^q} \sin(2\pi j x_1)\cos(2\pi j x_2)\right).$$

Here $\omega = (\omega_1,\dots,\omega_s) \in \mathbb{R}^s$ is such that $\omega_j \sim N(0,\sigma^2)$ for a fixed $\sigma > 0$. Random fields of this form are widely used, see [2], [23] and [6] for examples. Note that the right-hand side f is deterministic.

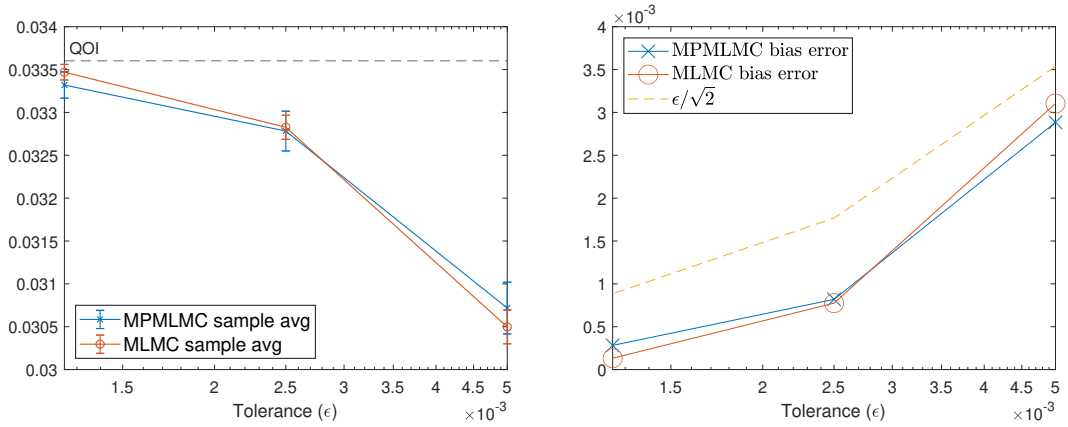
Compared to Section 5.1, one important difference occurs here. Due to the fact that $\omega_j \sim N(0,\sigma^2)$, Assumptions 1.12 and 1.14 are not satisfied anymore. This could be resolved using, for example, $\omega_j \sim \mathrm{Uni}(0,c)$, in which case both assumptions are satisfied as in the previous section. However, we want to test the developed methods under more realistic circumstances. Let us therefore discuss in more details the implications of the fact that Assumptions 1.12 and 1.14 do not hold. The classical approach is to replace Assumption 1.12 with the following: The random field a from the AVP with random data (Problem 1.11) satisfies that for a.e. $\omega \in \Omega$ there exist $a_{min}(\omega)$ and $a_{max}(\omega)$ such that for a.e. $x \in D$ it holds that

$$0 < a_{min}(\omega) \le a(x,\omega) \le a_{max}(\omega) < \infty. \tag{5.6}$$

Assumption 1.14 can be modified analogously. Moreover, further smoothness assumptions considering a, a_{min}, and a_{max} have to be made; see Assumptions A1 to A3 in [26].

Let us now discuss how this change affects the theory of the FEM and the Monte Carlo methods. It is possible to prove that under these modified assumptions all the error estimates in Section 1.3.3 hold with, ultimately, the constant C in Theorem 1.16 depending on ω. Consequently, the complexity theorems from Chapter 2 remain valid. A derivation of how the generic constant C in the error estimates depends on ω can be found in [26]. As noted in the aforementioned paper, the smoothness assumptions are satisfied for the elliptic problem with lognormal coefficients.

The other point which needs to be addressed is the impact of the modification of Assumptions 1.12 and 1.14 on the FE solution in finite precision arithmetic. To track down how the constants in the estimates are affected, it is necessary to go back to Lemma 1.18 where the underlying estimate is given. We will not discuss it here in detail. We only note that from (5.6) it follows that the coercivity constant α from (1.14) depends now on ω which leads to the constant C from Lemma 1.18 being dependent on ω. Lemma 1.18 is then used in Lemma 4.11 to verify the assumptions of the MPMLMC error theorem (Theorem 4.9). The proof of Lemma 4.11 has to be then modified in the way that under certain smoothness



(a) Sample average with 95% asymptotic confidence intervals.

(b) Bias error.

Figure 5.5: Comparison of MLMC using mixed precision and MLMC in reference precision (double) for an elliptic PDE with lognormal coefficients. MPMLMC achieves approximately the same error as the reference MLMC while using quarter and half precisions for the dominant part of the computations (the matrix factorization).

assumptions on the random field a, Hölder's inequality in Bochner spaces has to be employed to obtain the estimate (4.16). The general approach is similar to the estimates given in [26].

Our observations can be informally summarised as follows: If the random field a is sufficiently smooth and if Assumptions 1.12 and 1.14 are modified in the sense of (5.6) then all the theoretical results remain valid also for the equation (5.5) with the difference that the hidden constants in the asymptotic error estimates depend now on ω.

5.2.3 Numerical examples

In the first example of this section, we choose the data in (5.5) as follows: The right-hand side satisfies $f \equiv 1$ on D and the parameters in the coefficient function are chosen as $s = 4$, $q = 2$, and $\sigma^2 = 1.4$. To solve this problem numerically, we employ the same numerical methods as in Section 5.1. Both the variational and the discrete formulations are analogous as well. The only difference as far as the numerical solution is concerned is the fact that in this case the matrix decomposition cannot be precomputed, since the stiffness matrix A depends on the random parameter vector ω. In order to assemble the stiffness matrix, which is done using the element stiffness matrices, a numerical quadrature has to be employed. One of the standard choices is the midpoint rule; see [6]. We use the four-point Gaussian quadrature on each of the elements.

The adaptive MPMLMC algorithm (Algorithm 4) is run with the choice $k_p = 0.1$ for three values of the tolerance ϵ, namely $\epsilon = 5, 2.5, 1.25 \times 10^{-3}$. The parameters α, β, and γ from the MLMC complexity theorem (Theorem 2.10) are chosen as in the previous section ($\beta = 4$, $\alpha = 2$) except for the parameter γ. Here we choose $\gamma = 3$ based on the theoretical complexity of a sparse direct solver, which is in our case $O(n^{3/2})$ where n is the dimension of the matrix; see

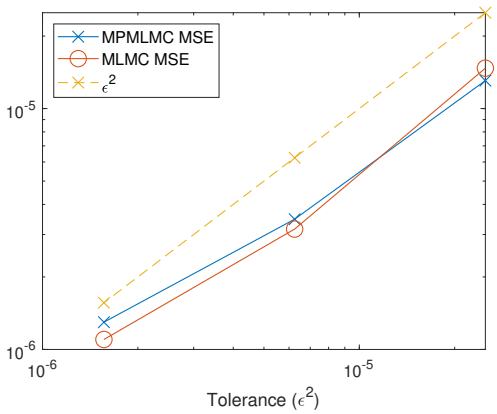


Figure 5.6: MSE of the MPMLMC and MLMC method — elliptic PDE with the lognormal coefficient. Both methods achieve approximately the same MSE in accordance with predictions and the error is safely below the given tolerance ϵ^2.

Section 3.2.1. The setting of iterative refinement is in this case the same as in the second example in the previous section, given by Table 5.2 with one difference, this being the fact that on the level $l = L = 3$ we use here the "*ssdd*" variant of iterative refinement. We run Algorithm 4 for each of the three values of the tolerance and show (Figure 5.5) the resulting sample average and bias error of the MPMLMC estimator (computed using 200 samples) along with the values for the reference MLMC estimator (using double precision, 500 samples). The reference MLMC estimator uses the same levels and number of samples on each level as the MPMLMC estimator. We observe that considering the sample average MPMLMC achieves approximately the same error as the reference MLMC while using quarter and half precision for a majority of the computations (i.e., the matrix factorization). We compute also the reference value of the quantity of interest. To this end we use Algorithm 1 using the tolerance $\epsilon = 2 \times 10^{-4}$ (this is the bounding value of the tolerance for which the computational time is reasonable using our setup). The reference value of the QOI is shown in Figure 5.5a.

Figure 5.6 shows the resulting bias error given by $|\mathbb{E}[\widehat{Q}^{\mathrm{MPML}}_{L,\{N_l\},\{\delta_l\}}] - \mathrm{QOI}|$ (or $|\mathbb{E}[\widehat{Q}^{\mathrm{ML}}_{L,\{N_l\}}] - \mathrm{QOI}|$ in case of the MLMC estimator). We observe that the bias error achieved by both estimators is approximately the same and the error is safely below the given tolerance so that the desired accuracy is achieved.

As noted above, the mixed precision setting is given by Table 5.2. This means that the dominant part of the computations, i.e., the matrix factorization, is carried out in quarter precision on the coarsest level and half precision on all the finer levels. Therefore, Corollary 4.13 gives us a 4–8× theoretical speedup in this nontrivial case of the elliptic PDE with a lognormal coefficient.

From our observations we can conclude that in many nontrivial cases and for the values of the tolerance which are of interest we can use quarter precision for the dominant part of the computations on the coarsest level and half precision for the dominant part of the computations on the remaining levels (apart from the finest one, possibly). In these cases the theoretical speedup of the computations is 4–8×. In the case where our problem is very ill-conditioned, we may be forced to switch to higher precision earlier and in such cases the speedup can be 4× or

2–4×, depending on the exact setting.

Note that we have not encountered overflow in the low precision matrix factorization in any of the examples presented in this chapter. In the case where overflow occurs, scaling or shifting techniques can be used, see [21].

Conclusion

This work makes three main contributions to existing Monte Carlo methods. The first major contribution is the rigorous analysis of Monte Carlo methods in finite precision arithmetic given via error estimates, which are fully general and can be straightforwardly applied to any PDE- or ODE-based problem as long as the assumptions of the respective theorems are satisfied. Namely, we have analysed the standard Monte Carlo method and the multilevel Monte Carlo method.

Our second major contribution involves the model problem, i.e., the elliptic PDE with random coefficients and a random right-hand side. To solve this model problem numerically, we employed the finite element method leading to a system of linear equations. Our analysis shows that if we are able to control the error of the solution of this linear system in the sense of the relative residual norm, we are able to apply the aforementioned general error estimates and thus to control the effect of finite precision arithmetic in Monte Carlo methods for this problem.

The third major contribution is connected to the proposed adaptive mixed precision MLMC algorithm. The optimal values of precision can be computed "on the fly" with no additional cost and the theoretical error estimates ensure that the overall error is below a given tolerance. The adaptive algorithm can be straightforwardly employed when any direct or iterative solver is used for solving the underlying linear system. We choose to discuss in detail the case when a direct solver is used. The adaptive algorithm has been extensively tested using a direct solver with iterative refinement on various settings of the model problem including Poisson's equation with a random right-hand side and an elliptic PDE with lognormal random coefficients, achieving a theoretical speedup of 4–8$\times$ compared to the reference double precision. The modeled speedup is achieved under the assumption that when single, half, or quarter precision is used instead of double precision, the runtime is improved by a factor of 2, 4 or 8, respectively.

We have also extended the existing analysis of iterative refinement to linear systems stemming from the finite element method to better understand the behaviour of iterative refinement in the context of our model problem. We have verified our theoretical results regarding the MC methods on numerous examples, one of the conclusions being the fact that we have successfully used quarter precision in a non-trivial example which appears to be a rare achievement within scientific computing applications.

The presented results offer many possibilities for further research, such as the use of mixed precision within other uncertainty quantification algorithms, for example, the multi-index Monte Carlo method [17] or the multilevel Markov chain Monte Carlo method [9]. Further gain from using mixed precision in the MLMC method can be obtained if the sampling and the discretisation error are not balanced equally but rather they are balanced optimally as in the continuation MLMC method [7]. We see this as another promising possibility for further research.

Bibliography

[1] Ahmad Abdelfattah, Hartwig Anzt, Erik G Boman, Erin Carson, Terry Cojean, Jack Dongarra, Alyson Fox, Mark Gates, Nicholas J Higham, Xiaoye S Li, et al. A survey of numerical linear algebra methods utilizing mixed-precision arithmetic. *The International Journal of High Performance Computing Applications*, 35(4):344–369, 2021.

[2] Ivo Babuška, Raúl Tempone, and Georgios E Zouraris. Galerkin finite element approximations of stochastic elliptic partial differential equations. *SIAM Journal on Numerical Analysis*, 42(2):800–825, 2004.

[3] Grey Ballard, James Demmel, Olga Holtz, and Oded Schwartz. Minimizing communication in numerical linear algebra. *SIAM Journal on Matrix Analysis and Applications*, 32(3):866–901, 2011.

[4] Christian Brugger, Christian de Schryver, Norbert Wehn, Steffen Omland, Mario Hefter, Klaus Ritter, Anton Kostiuk, and Ralf Korn. Mixed precision multilevel Monte Carlo on hybrid computing systems. In *2014 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr)*, pages 215–222. IEEE, 2014.

[5] Erin Carson and Nicholas J Higham. Accelerating the solution of linear systems by iterative refinement in three precisions. *SIAM Journal on Scientific Computing*, 40(2):A817–A847, 2018.

[6] K Andrew Cliffe, Mike B Giles, Robert Scheichl, and Aretha L Teckentrup. Multilevel Monte Carlo methods and applications to elliptic PDEs with random coefficients. *Computing and Visualization in Science*, 14:3–15, 2011.

[7] Nathan Collier, Abdul-Lateef Haji-Ali, Fabio Nobile, Erik Von Schwerin, and Raúl Tempone. A continuation multilevel Monte Carlo algorithm. *BIT Numerical Mathematics*, 55:399–432, 2015.

[8] James Demmel, Yozo Hida, William Kahan, Xiaoye S Li, Sonil Mukherjee, and E Jason Riedy. Error bounds from extra-precise iterative refinement. *ACM Transactions on Mathematical Software (TOMS)*, 32(2):325–351, 2006.

[9] Tim J Dodwell, Christian Ketelsen, Robert Scheichl, and Aretha L Teckentrup. A hierarchical multilevel Markov chain Monte Carlo algorithm with applications to uncertainty quantification in subsurface flow. *SIAM/ASA Journal on Uncertainty Quantification*, 3(1):1075–1108, 2015.

[10] Alexandre Ern and Jean-Luc Guermond. *Theory and practice of finite elements*, volume 159. Springer, 2004.

[11] R Allan Freeze. A stochastic-conceptual analysis of one-dimensional groundwater flow in nonuniform homogeneous media. *Water resources research*, 11(5):725–741, 1975.

[12] Alexander D Gilbert, Ivan G Graham, Frances Y Kuo, Robert Scheichl, and Ian H Sloan. Analysis of quasi-Monte Carlo methods for elliptic eigenvalue problems with stochastic coefficients. *Numerische Mathematik*, 142:863–915, 2019.

[13] Michael B Giles. Multilevel Monte Carlo path simulation. *Operations research*, 56(3):607–617, 2008.

[14] Michael B Giles. Multilevel Monte Carlo methods. *Acta numerica*, 24:259–328, 2015.

[15] Pierre Grisvard. *Elliptic problems in nonsmooth domains*. SIAM, 2011.

[16] Azzam Haidar, Stanimire Tomov, Jack Dongarra, and Nicholas J Higham. Harnessing GPU tensor cores for fast FP16 arithmetic to speed up mixed-precision iterative refinement solvers. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 603–613. IEEE, 2018.

[17] Abdul-Lateef Haji-Ali, Fabio Nobile, and Raúl Tempone. Multi-index Monte Carlo: when sparsity meets sampling. *Numerische Mathematik*, 132:767–806, 2016.

[18] Nicholas J Higham. Chop - MATLAB code for rounding matrix elements to lower precision. URL: https://github.com/higham/chop. [Accessed 23-4-2023].

[19] Nicholas J Higham. *Accuracy and stability of numerical algorithms*. SIAM, 2002.

[20] Nicholas J Higham and Theo Mary. Mixed precision algorithms in numerical linear algebra. *Acta Numerica*, 31:347–414, 2022.

[21] Nicholas J Higham, Srikara Pranesh, and Mawussi Zounon. Squeezing a matrix into half precision, with an application to solving linear systems. *SIAM journal on scientific computing*, 41(4):A2536–A2551, 2019.

[22] Robert J Hoeksema and Peter K Kitanidis. Analysis of the spatial structure of properties of selected aquifers. *Water resources research*, 21(4):563–572, 1985.

[23] Fabio Nobile, Raúl Tempone, and Clayton G Webster. A sparse grid stochastic collocation method for partial differential equations with random input data. *SIAM Journal on Numerical Analysis*, 46(5):2309–2345, 2008.

[24] Anonymous report. NVIDIA H100 Tensor Core GPU Architecture. URL: https://resources.nvidia.com/en-us-tensor-core. [Accessed 23-4-2023].

[25] Adrian E Scheidegger. *The physics of flow through porous media*. University of Toronto press, 1957.

[26] Aretha L Teckentrup, Robert Scheichl, Michael B Giles, and Elisabeth Ullmann. Further analysis of multilevel Monte Carlo methods for elliptic PDEs with random coefficients. *Numerische Mathematik*, 125:569–600, 2013.

[27] Bastien Vieuble. *Mixed precision iterative refinement for the solution of large sparse linear systems.* PhD thesis, INP Toulouse, 2022.

List of Figures

List of Tables