CHALMERS
UNIVERSITY OF TECHNOLOGY

Computing Failure Probabilities for PDEs with Random Data

Master's thesis in Engineering Mathematics and Computational Science

OSKAR EKLUND

Department of Mathematical Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Master's thesis 2020:NN

Computing Failure Probabilities for PDEs with Random Data

OSKAR EKLUND

Department of Mathematical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2020

Computing Failure Probabilities for PDEs with Random Data
Oskar Eklund

Supervisor: Axel Målqvist, Mathematical Sciences
Ass. supervisor: Fredrik Hellman, Mathematical Sciences
Examiner: Annika Lang, Mathematical Sciences

Master's Thesis 2020:NN
Department of Mathematical Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2020

Computing Failure Probabilities for PDEs with Random Data
Oskar Eklund
Mathematical Sciences
Chalmers University of Technology

Abstract

The thesis deals with partial differential equations with random data and in particular Poisson's equation with random data. This equation has a unique solution. The failure probability is the probability that a functional of that solution is less (or greater) than a given value. Algorithms for approximating failure probabilities are studied and tested and a new iterative method of approximating the failure probability is presented and examined in numerical experiments. As the thesis involves both random variables and partial differential equations, both probabilistic problems and problems with partial differential equations are studied along the way. The results of the numerical experiments shows that the method performed well, with respect to computational cost, in comparison with a basic Monte Carlo simulation.

Keywords: PDEs with random data, failure probability, Monte Carlo, finite element method, selective refinement

Acknowledgements

.....later.....

Oskar Eklund, Gothenburg, August 2020

Contents

List of Figures **xi**

1 Introduction **1**

2 Theoretical background and problem formulation **3**
2.1 Preliminaries 3
2.2 Problem formulation 3

3 Finite element method **7**
3.1 Preliminaries 7
3.2 Variational formulation 8
3.3 Discretization 9
3.4 Error analysis 10

4 Probability theory **13**
4.1 Generating data 13
4.2 Monte Carlo method 15

5 Algorithms **17**
5.1 Selective refinement algorithm 17
5.2 Variation of the selective refinement algorithm 20

6 Results **25**
6.1 A distribution example 25
6.2 The maximum 26
6.3 Discussion 27

Bibliography **29**

List of Figures

2.1 Realizations of a 4

4.1 Plot of the covariance matrix C associated with covariance function $C(x_i, x_j) = \mathbb{E}[a(x_i)a(x_j)] = \sigma^2 e^{\frac{-\|x_i - x_j\|_2}{\rho}}$, $\sigma = 1$, $\rho = 1$ and a uniform grid of 20×20 points. The matrix is BTTB with 20×20 Toeplitz blocks of size 20×20. 15

5.1 Illustration of (5.3) 18
5.2 An illustration of one iteration of Algorithm 3 with $L = 4$, $N_0 = 10$, $N_1 = 5$, $N_2 = 3$, $N_3 = 2$, $N_4 = 1$. The blue dots are to be refined and the red dots are finished refined. 22

6.1 Solution u for problem (2.1) 26
6.2 The triangulations of D used for level $\ell = 0$ and $\ell = 1$ 27

1
Introduction

Partial differential equations (PDEs) are widely used for describing physical phenomena. In this project, we take a deeper look into PDEs with random data, that is, PDEs with stochastic coefficients. With the work of [1] and [2] in mind, we study algorithms for efficiently computing approximations of failure probabilities arising from such PDEs. In the project, we have developed an algorithm for making the computation of a failure probability efficient. We present a new method for computing the failure probability and show two numerical examples where the method, consisting of two algorithms, are used.

The main problem of this thesis is to, given a convex domain $D \subset \mathbb{R}^2$, a probability space $(\Omega, \mathcal{F}, P)$, a log-normally distributed random field a and $\omega \in \Omega$, find a function u such that

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

for a fixed $f \in L^2(D)$. This model equation has been studied in e.g. [3]. It is a frequently used model in porous media flow. In our project we deal with efficient ways of approximating failure probabilities for functionals of the solution (the probability of a functional to be greater or less than some value) to (1.1).

In [1] and [2] a selective refinement algorithm together was introduced and used together with a multilevel Monte Carlo method for approximating failure probabilities. Here, instead of a selective refinement algorithm, we suggest a variation of this approach which leads to an iterative algorithm that does not demand for error estimates as in [1] and [2]. We use the circulant embedding method for generating the data a and the standard Monte Carlo method.

In Chapter 2 we start by some preliminary definitions followed by formulating the problem of the thesis. This is followed by two independent chapters; Chapter 3 which contains the finite element theory used for solving PDEs and Chapter 4 which contains the probabilistic part of the project. Chapter 5 describes the algorithms which form the method used and Chapter 6 presents the results of the implementation of the method. Also, a discussion with conclusions and proposals of future work will be included in Chapter 6, which ends the thesis.

2

Theoretical background and problem formulation

In this chapter, we present the theoretical background of the thesis and the problem we are dealing with. The aim of this chapter is to make the reader familiar with the theory behind the algorithms and the numerical results.

2.1 Preliminaries

We start by presenting some useful definitions. First, we define a random field.

Definition 2.1.1. Let T be a set and $(E, \mathcal{E})$ be a measurable space. A *random field* $(a(t),\, t \in T)$ taking values in $(E, \mathcal{E})$ is a collection of measurable mappings $a(t)$ from a probability space $(\Omega, \mathcal{F}, \mathbb{P})$.

For us, T will be a subset of $\mathbb{R}^2$, E will be $\mathbb{R}$ and $\mathcal{E}$ will be the Borel σ-algebra containing all subsets of $\mathbb{R}$, denoted by $\mathcal{B}(\mathbb{R})$. With this definition in mind, we continue with defining a normally distributed random field and then a log-normally distributed random field, which will be of certain interest in the main problem of our project.

Definition 2.1.2. A random field $(a(t),\, t \in T)$ is *normally distributed* if for all $n \geq 1$ and $(t_1, \ldots, t_n) \in T^n$ the random vector $(a(t_1), \ldots, a(t_n))$ is a Gaussian random vector.

Worth mentioning here is that we by a *Gaussian random vector* mean a random vector, where every finite linear combination of its components is a Gaussian random variable.

Definition 2.1.3. A random field $(A(t),\, t \in T)$ is *log-normally distributed* if the random field $(\ln(a(t)), t \in T)$ is normally distributed.

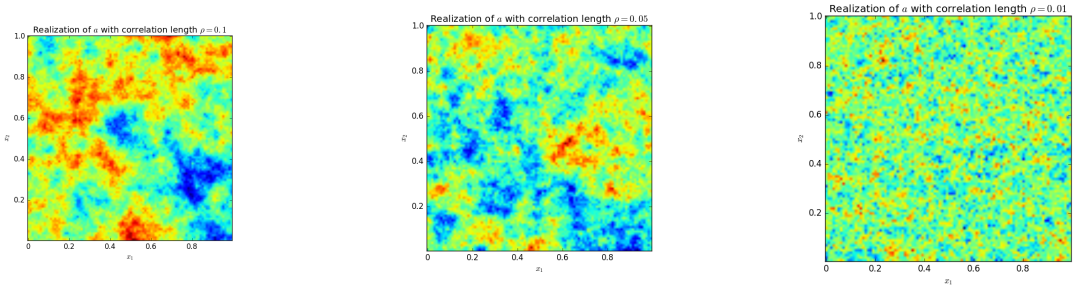
2.2 Problem formulation

Now we are ready for formulating the problem of the thesis. We are considering Poisson's equation with random data. Let D be a convex domain, $(\Omega, \mathcal{F}, \mathbb{P})$ a

probability space and a a log-normally distributed random field in D ($T = D$ in Definition 2.1.3). Consider the problem of, given $\omega \in \Omega$, finding a function u such that

$$\begin{cases} -\nabla \cdot a(\omega, x)\nabla u(\omega, x) &= f(x), \quad \text{in } D, \\ u(\omega, x) &= 0, \quad \text{on } \partial D, \end{cases} \tag{2.1}$$

for a fixed $f \in L^2(D)$. Here, $a(\omega, x) = e^{\kappa(\omega,x)}$, with $\kappa(\cdot, x)$ being a normally distributed random field with mean $\mathbb{E}[\kappa(\cdot, x)] = 0$ and covariance function given by $\mathbb{E}[\kappa(\cdot, x_1)\kappa(\cdot, x_2)] = \sigma^2 e^{\frac{-\|x_1 - x_2\|_2}{\rho}}$. The norm $\|\cdot\|_2$ denotes the Euclidean norm and the given parameters σ, $\rho \in \mathbb{R}^+$ are called *standard deviation* and *correlation length*, respectively.



(a) $\rho = 0.1$ **(b)** $\rho = 0.05$ **(c)** $\rho = 0.01$

Figure 2.1: Realizations of a

In the figures above we see how realizations of a may look like for standard deviation $\sigma = 1$ and different correlation lengths ρ. The first question one may ask while looking at (2.1) is if it admits a unique solution. For this purpose, we look into the work of [3].

Theorem 2.2.1. *Equation (2.1) admits a unique solution u, which belongs to $L^p(\Omega, H_0^1(D))$, for any $p > 0$.*

For a proof of Theorem 2.2.1, we refer to Proposition 2.4 of [3]. We proceed by defining the failure probability of a functional of the solution u to (2.1), which is the term of most interest in this project. The failure probability should be seen as the probability that a given quantity of interest (QoI), expressed in terms of a functional X of u, is less (or greater) than a given critical value.

Definition 2.2.1. Let u be the unique solution to (2.1), X a functional of u and $y \in \mathbb{R}$. Then the *failure probability* p of X is given by

$$p = \mathbb{P}(\{X \leq y\}). \tag{2.2}$$

For clarity, we mention here that as u is a random variable, also X becomes a random variable. Thus, it makes sense to talk about the event $\{X \leq y\}$ and the

probability $\mathbb{P}(\{X \leq y\})$. We follow up with an example which shows how failure probabilities relate to real world problems, inspired by [1].

Example 2.2.1. Geological sequestration of carbon dioxide (CO_2) is performed by injection of CO_2 in an underground reservoir. The fate of the CO_2 determines the success or failure of the storage system. The CO_2 propagation is often modeled as a PDE with random input data, such as a random permeability field. Typical QoIs include reservoir breakthrough time or pressure at a fault. The value y corresponds to a critical value which the QoI may not exceed or go below. In the breakthrough time case, low values are considered failures. In the pressure case, high values are considered failures. In the latter case, one should negate the QoI to transform the problem into the form of (2.2).

To compute failure probabilities, we consider the (Bernoulli distributed) random variable Q, defined by $Q = \mathbb{1}_{\{X \leq y\}}$ and note that $p = \mathbb{E}[Q]$. Later on, we construct estimators $\hat{Q}$ to approximate $\mathbb{E}[Q]$.

3

Finite element method

In this chapter we consider a deterministic version of (2.1); the boundary value problem

$$\begin{cases} -\nabla \cdot A(x)\nabla u(x) &= f(x), \quad \text{in } D, \\ u(x) &= 0, \qquad \text{on } \partial D, \end{cases} \tag{3.1}$$

where $D \subset \mathbb{R}^2$ is a polygonal domain and where we for the diffusion coefficient $A \in L^\infty(D, \mathbb{R}^{2\times 2})$ assume it holds

$$0 < a_0 := \operatorname*{ess\,inf}_{x\in D} \inf_{v\in\mathbb{R}^2\setminus\{0\}} \frac{(A(x)v)\cdot v}{v\cdot v},$$
$$\infty > a_1 := \operatorname*{ess\,sup}_{x\in D} \sup_{v\in\mathbb{R}^2\setminus\{0\}} \frac{(A(x)v)\cdot v}{v\cdot v}.$$

We will present the variational formulation and proceed by introducing the finite element method for (3.1). For details on the finite element method, see e.g. [10].

3.1 Preliminaries

In this section we present preliminaries required to follow the rest of the chapter. We define the spaces $L^p(D)$ by

$$L^p(D) := \{u : D \to \mathbb{R} \text{ measurable } : \|u\|_{L^p(D)} < \infty\}, \text{ for } p = 1, 2,$$

with norm

$$\|u\|_{L^p(D)} := \left(\int_D |u(x)|^p \, \mathrm{d}x\right)^{1/p}.$$

The space $L^2(D)$ is a Hilbert space with respect to the inner product

$$(u, v) := \int_D u(x)v(x) \, \mathrm{d}x.$$

To get to the Sobolev spaces used we here introduce the concept of weak derivatives. Let $u \in C_0^1(D)$. For all $v \in C_0^1(D)$ we have

$$\int_D u\frac{\partial v}{\partial x_i} \, \mathrm{d}x = -\int_D v\frac{\partial u}{\partial x_i} \, \mathrm{d}x, \quad i = 1, 2.$$

Now we, for the partial derivatives, introduce the notation

$$D^\alpha v := \frac{\partial^{|\alpha|} v}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2}},$$

where $\alpha = (\alpha_1, \alpha_2)$ is a multi-index and $|\alpha| = \alpha_1 + \alpha_2$.

Definition 3.1.1. If $u, v \in L^1(D)$ we say that v is the α^{th}*-weak derivative* of u if

$$\int_D u D^\alpha \varphi \ \mathrm{d}x = (-1)^{|\alpha|} \int_D v\varphi \ \mathrm{d}x, \quad \text{for all } \phi \in C_0^{|\alpha|}(D).$$

We note that it coincides with the (strong) derivative for $u \in C_0^1(D)$. Now we let

$$H^k(D) := \{v \in L^2(D) : \ D^\alpha v \in L^2(D) \text{ for } |\alpha| \leq k\}, \ \text{ for } k = 1, 2. \tag{3.2}$$

Also, we let $(v, w)_{H^k(D)} := \sum_{|\alpha| \leq k} \int_d D^\alpha v D^\alpha w \ \mathrm{d}x$ and

$$\|v\|_{H^k(D)} := (v, w)_{H^k(D)}^{1/2} = \left(\sum_{|\alpha| \leq k} \int_D D^\alpha v D^\alpha w \ \mathrm{d}x \right)^{1/2}. \tag{3.3}$$

The set (3.2) with norm (3.3) is an example of a Sobolev space, which is a Banach space because of the weak derivatives. We also use the notation

$$H_0^1(D) := \{v \in H^1(D) : v|_{\partial D} = 0\}.$$

We continue with the Cauchy–Schwarz inequality.

Theorem 3.1.1. *Let V be a vector space with inner product $(\cdot, \cdot)$ and norm $\| \cdot \|$. Then*

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

Also, we have the important Lax–Milgram theorem which gives existence and uniqueness (in weak sense) of a solution to a given boundary value problem.

Theorem 3.1.2. *Let V be a Hilbert space and assume a to be a coercive, bounded bilinear form and L a bounded linear functional on V. Then, there exists a unique $u \in V$ such that*

$$a(u, v) = L(v), \quad \textit{for all } v \in V.$$

3.2 Variational formulation

To get to the variational formulation of (3.1), we multiply the equation with a test function $v \in H_0^1(D)$. Also, we integrate over D, use Green's formula and the boundary conditions. The variational formulation of (3.1), reads: find $u \in H_0^1(D)$ such that

$$(A\nabla u, \nabla v) = (f, v) \qquad \text{for all } v \in H_0^1(D), \tag{3.4}$$

where $(\cdot, \cdot)$ denotes the L^2 inner product.

Theorem 3.2.1. *There exists a unique function $u \in H_0^1(D)$ fulfilling (3.4).*

Proof. We equip $H_0^1(D)$ with the norm $|\cdot|_{H^1(D)} := \|\nabla \cdot\|_{L^2(D)}$. This is a norm on $H_0^1(D)$ because of the Poincaré inequality. We want to apply the Lax–Milgram theorem. For that purpose, we have to show that the left-hand side of (3.4) is bounded and coercive and that the right-hand side of (3.4) is bounded. Let $u, v \in H_0^1(D)$. We have

$$|(A\nabla u, \nabla v)| \leq a_1 |u|_{H^1(D)} |v|_{H^1(D)},$$

by the Cauchy–Schwarz inequality, which proves the boundedness of the left-hand side. Also,

$$\begin{aligned}(A\nabla u, \nabla u) &\geq a_0(\nabla u, \nabla u) \\ &= a_0 |u|^2_{H^1(D)},\end{aligned}$$

which proves the coercivity of the left-hand side. Finally, by the Poincaré inequality, there exists a constant C such that

$$\begin{aligned}|(f, v)| &\leq \|f\|_{L^2(D)} \|v\|_{L^2(D)} \\ &\leq C \|f\|_{L^2(D)} |v|_{H^1(D)}.\end{aligned}$$

The first inequality is, once again, by the Cauchy–Schwarz inequality. Now we have boundedness also for the right-hand side, and the Lax–Milgram theorem gives us the result. □

As we have the variational formulation of (3.1), we will make a discretization to get to the finite element problem.

3.3 Discretization

First we let $\mathcal{T}_h = \{K\}$ be a triangulation of D and let $h = \max_{K \in \mathcal{T}_h}(\operatorname{diam}(K))$. We assume the triangulation to be *shape-regular*, i.e., there exists a number $\epsilon > 0$ such that every $K \in \mathcal{T}_h$ contains a ball of radius r_ϵ with $r_\epsilon \geq \frac{h_K}{\epsilon}$, where $h_K = \operatorname{diam}(K)$. Also, we let V_h be the space of continuous piecewise linear polynomials on $\mathcal{T}_h$, with trace 0, i.e.,

$$V_h = \{v \in \mathcal{C}(D) : \ v \text{ is affine on } K \text{ for each } K \in \mathcal{T}_h, \ v = 0 \text{ on } \partial D\}.$$

Now our finite element problem reads: find $u_h \in V_h$ such that

$$(A\nabla u_h, \nabla v) = (f, v) \qquad \text{for all } v \in V_h. \tag{3.5}$$

Theorem 3.3.1. *There exists a unique finite element solution $u_h \in V_h$ to (3.5).*

Proof. Since V_h is a closed subspace of $H_0^1(D)$ the result follows as in the proof of Theorem 3.1.1. □

Next, we want to get an idea of the error that occurs while approximating u by u_h, in terms of h.

3.4 Error analysis

For the purpose of getting error estimates in terms of h, we want to introduce the *Clément interpolant*. Let $\mathcal{N}$ denote the nodes of $\mathcal{T}_h$ and $\mathbf{x}_i$ be the coordinates of node i. First, we define the *local patches*

$$w_i = \bigcup \{K \in \mathcal{T}_h : \ x_i \in K\}$$

and

$$\tilde{w}_K = \bigcup \{w_i : \ K \in w_i\}.$$

Let $\{\phi_i\}_{i\in\mathcal{N}}$ be the set of hat functions spanning V_h. We are ready for the definition of the Clément interpolant.

Definition 3.4.1. The *Clément interpolant* $\mathcal{I}_h : \ L^1(D) \to V_h$ is, for $v \in L^1(D)$, given by

$$\mathcal{I}_h v = \sum_{i\in\mathcal{N}} \frac{\int_{w_i} v \ \mathrm{d}x}{\int_{w_i} 1 \ \mathrm{d}x}\phi_i.$$

We continue with a interpolation estimate.

Lemma 3.4.1. *For all functions $v \in H^{1+s}(D)$, for $s = 0, 1$, it holds for some constant C*

$$\|v - \mathcal{I}_h v\|_{L^2(K)} + h_K\|v - \mathcal{I}_h v\|_{H^1(K)} \leq Ch_K^{1+s}\|v\|_{H^{1+s}(\tilde{w}_K)}.$$

Further, due to the shape-regularity of $\mathcal{T}_h$, it holds for some constant C

$$\|v\|^2_{H^{1+s}(D)} \leq \sum_{K\in\mathcal{T}_h} \|v\|^2_{H^{1+s}(\tilde{w}_K)} \leq C\|v\|^2_{H^{1+s}(D)}.$$

For a proof of Lemma 3.3.1, we refer to Section 6.9 in Chapter II of [5]. As we have this estimate, we are ready for the next lemma. To make the reading easier in the upcoming proofs, we let $b(u, v) = (A\nabla u, \nabla v)$ for $u, v \in V_h$.

Lemma 3.4.2 (Céa's lemma)**.** *Assume u and u_h solve (3.4) and (3.5), repsectively. Then, there exists a constant C such that*

$$\|u - u_h\|_{H^1(D)} \leq C \min_{\chi\in V_h} \|u - \chi\|_{H^1(D)}. \tag{3.6}$$

Proof. For any $\chi \in V_h$ we have, by (3.4) and (3.5),

$$\begin{aligned} b(u - u_h, \chi - u_h) &= (f, \chi - u_h) - (f, \chi - u_h) \\ &= 0. \end{aligned}$$

Now, using that the bilinear form b is coercive and bounded we get, with coercivity constant α and boundedness constant M,

$$\begin{aligned} \alpha\|u - u_h\|^2_{H^1(D)} &\leq b(u - u_h, u - u_h) \\ &\leq b(u - u_h, u - \chi) + b(u - u_h, \chi - u_h) \\ &\leq M\|u - u_h\|_{H^1(D)}\|u - \chi\|_{H^1(D)}. \end{aligned}$$

Hence, with $C = \frac{M}{\alpha}$,

$$\|u - u_h\|_{H^1(D)} \leq C \min_{\chi \in V_h} \|u - \chi\|_{H^1(D)}.$$

□

In the next lemma, we combine Lemma 3.1.1. and Lemma 3.1.2. to get to a new estimate.

Lemma 3.4.3. *Assume $u \in H^2(D)$ and u_h solve (3.4) and (3.5), respectively. Then, there exists a constant C such that*

$$\|u - u_h\|_{H^1(D)} \leq Ch\|u\|_{H^2(D)}. \tag{3.7}$$

Proof. We use Lemma 3.2.2. with $\chi = \mathcal{I}_h u$ (in the first inequality) and the two estimates of Lemma 3.2.1 with $s = 1$ (in the second inequality) to conclude

$$\begin{aligned} \|u - u_h\|_{H^1(D)} &\leq C\|u - \mathcal{I}_h u\|_{H^1(D)} \\ &\leq Ch\|u\|_{H^2(D)}. \end{aligned}$$

□

Now we want to bound the error in the $L^2(D)$-norm.

Theorem 3.4.1. *Assume u and u_h solve (3.4) and (3.5), repsectively. Assume also that there exists a constant C_0 such that $||v||_{H^2(D)} \leq C_0|| - \nabla \cdot A\nabla v||_{L^2(D)}$ for all $v \in H^2(D) \cap H_0^1(D)$. Let the functional $q : V_h \to \mathbb{R}$ be defined by*

$$q(v) = \int_D v\psi \ \mathrm{d}x,$$

for some $\psi \in L^2(D)$. Then, there exists a constant C such that

$$q(u - u_h) \leq Ch^2 \tag{3.8}$$

Proof. We introduce the adjoint problem

$$\begin{cases} -\nabla \cdot A(x)\nabla\varphi(x) &= \psi(x), \quad \text{in } D, \\ \varphi(x) &= 0, \qquad \text{on } \partial D. \end{cases}$$

In weak formulation, it reads: find $\varphi \in H^2(D) \cap H_0^1(D)$ such that

$$b(v, \varphi) = (v, \psi) \quad \text{for all } v \in V_h. \tag{3.9}$$

Now, we get

$$\begin{aligned} q(u - u_h) &= b(u - u_h, \varphi) \\ &= b(u - u_h, \varphi - \mathcal{I}_h\varphi) \\ &\leq M\|u - u_h\|_{H^1(D)}\|\varphi - \mathcal{I}_h\varphi\|_{H^1(D)} \\ &\leq Mh^2\|u\|_{H^2(D)}\|\varphi\|_{H^2(D)} \\ &\leq MC_0^2h^2\| - \nabla \cdot A\nabla u\|_{L^2(D)}\| - \nabla \cdot A\nabla\varphi\|_{L^2(D)} \\ &= MC_0^2\|f\|_{L^2(D)}\|\psi\|_{L^2(D)}h^2 \end{aligned} \tag{3.10}$$

To be clear: in the first equality we used (3.9) with $v = u - u_h$ and in the second equality we used that $b(u - u_h, \mathcal{I}_h\varphi) = 0$. Further, in the third line we used the boundedness of b with M being the boundedness constant. Then, in the 4^{th} line, we use Lemma 4.2.3 twice. In the 5^{th} line, we use directly an assumption of the theorem. Hence, we can conlude that there exists a constant C such that

$$q(u - u_h) \leq Ch^2.$$

□

Choosing $\psi = u - u_h$ in Theorem 3.3.1. we obtain the following.

Corollary 3.4.1. *Assume u and u_h solve (3.4) and (3.5), repsectively. Assume also that there exists a constant C_0 such that $\|v\|_{H^2(D)} \leq C_0\| - \nabla \cdot A\nabla v\|_{L^2(D)}$ for all $v \in H^2(D) \cap H_0^1(D)$. Then, there exists a constant C such that*

$$\|u - u_h\|_{L^2(D)} \leq Ch^2. \tag{3.11}$$

This estimate will be of particular interest for the upcoming method.

4

Probability theory

The first probabilistic problem of the project is the problem of generating normally distributed random fields, needed for (2.1). The second is the problem of approximating the expectation of random variables. Approximations of expectations is often done by means of Monte Carlo methods, and so, it is done in this work. This chapter aims to present first how we are deling with generating the data and then the Monte Carlo method.

4.1 Generating data

For generating normally distributed random fields we make use of the *circulant embedding method* described in [4] to get a realization of a. We will in this section follow the work of [4]. For further details on the circulant embedding method, see [7].

Here we consider a two-dimensional domain $D \subset \mathbb{R}^2$. A normally distributed random field $\{a(x) : \ x \in D\}$ is a second-order field such that the vector of random variables

$$\mathbf{a} = [a(x_1), a(x_2), \ldots, a(x_N)]^T$$

follows the multivariate normal distribution for any $x_1, \ldots, x_N \in D$. This is written $\mathbf{a} \sim N(\mu, C)$, where μ is the mean vector with entries $\mu_i = \mu(x_i)$ and C is the covariance matrix with entries $c_{ij} = C(x_i, x_j)$. The covariance matrix C is by definition symmetric and non-negative definite. Further, we assume $\mu = \mathbf{0}$. An observation here is that a *pair* of independent samples from $N(0, C)$ can be drawn simultaneously by taking the real and imaginary parts of $Y \sim CN(\mathbf{0}, 2C)$, where CN denotes the complex normal distribution.

An $N \times N$ matrix is said to be a *Toeplitz* matrix if the entries along each diagonal are the same. Also, an $N \times N$ matrix is said to be a *circulant* matrix if it is a Toeplitz matrix for which each column is a circular shift of the elements in the preceding column (so that the last entry becomes the first entry). Now, consider the matrices

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 2 \\ 3 & 2 & 1 \end{pmatrix}, \qquad \begin{pmatrix} 1 & 3 & 3 \\ 3 & 1 & 3 \\ 3 & 3 & 1 \end{pmatrix}.$$

The matrix on the left is symmetric and Toeplitz while the matrix on the right is symmetric and circulant. We notice that symmetric Toeplitz matrices always can

be extended to symmetric circulant matrices by adding an extra row and column. For the symmetric Toeplitz matrix above it would look like this:

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 2 \\ 3 & 2 & 1 \end{pmatrix} \quad \longrightarrow \quad \begin{pmatrix} 1 & 2 & 3 & 2 \\ 2 & 1 & 2 & 3 \\ 3 & 2 & 1 & 2 \\ 2 & 3 & 2 & 1 \end{pmatrix}.$$

We notice further that we do only need to store the first row or column of a symmetric Toeplitz or circulant matrix to generate the whole matrix. Similarly to this extension of the above matrix, it is always possible to extend symmetric block Toeplitz matrices with Toeplitz blocks (BTTB matrices) to symmetric block circulant matrices with circulant blocks (BCCB matrices).

Circulant and BCCB matrices can be factorized using discrete Fourier transforms. Any BCCB matrix B has the decomposition $B = F\Lambda F^*$ where F is the two-dimensional Fourier matrix, F^* is the conjugate transpose of F and Λ is the diagonal matrix of eigenvalues. If B now is a valid covariance matrix (symmetric and non-negative definite), then samples can be drawn from $CN(\mathbf{0}, 2B)$. Then, since non of the eigenvalues of B are negative, $\Lambda^{1/2}$ is well-defined and we can compute

$$\mathbf{Y} = W\Lambda^{1/2}\xi, \qquad \text{where } \xi \sim CN(\mathbf{0}, 2I).$$

It can be shown that $\mathbf{Y} \sim CN(0, 2B)$ and so $\mathbf{Z}_1 = Re(\mathbf{Y})$ and $\mathbf{Z}_2 = Im(\mathbf{Y})$ are independent $N(\mathbf{0}, B)$ samples. The multiplications with W can be performed by applying the discrete Fourier transform and Λ can be obtained by applying an inverse discrete Fourier transform to the reduced version of B. The observations above are the basis for the circulant embedding method.

The $N \times N$ covariance matrix C associated with a normally distributed random field on a two-dimensional domain D is in general *not* a BCCB matrix. However, if the sample points x_i are uniformly spaced and the covariance function are stationary (as in our case), then C is a BTTB matrix. Consider now the covariance function

$$C(x_i, x_j) = \sigma^2 e^{\frac{-\|x_i - x_j\|_2}{\rho}}.$$

This is the logarithm of the covariance function of the random field in the main problem of the thesis. The Toeplitz structure is visible.

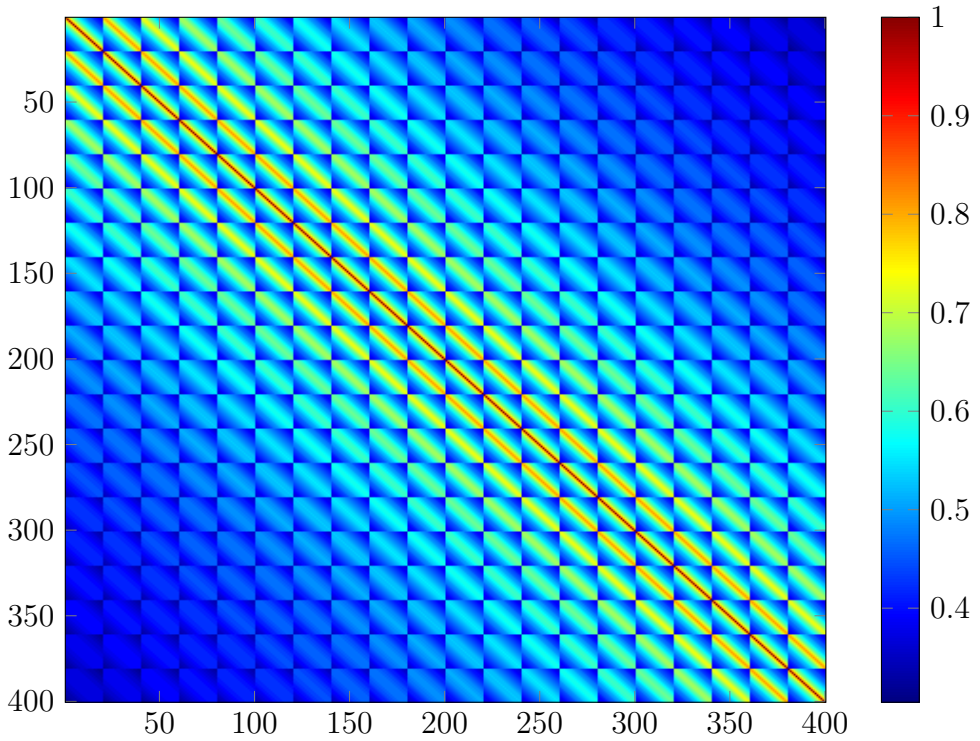


Figure 4.1: Plot of the covariance matrix C associated with covariance function $C(x_i, x_j) = \mathbb{E}[a(x_i)a(x_j)] = \sigma^2 e^{\frac{-\|x_i - x_j\|_2}{\rho}}$, $\sigma = 1$, $\rho = 1$ and a uniform grid of 20×20 points. The matrix is BTTB with 20×20 Toeplitz blocks of size 20×20.

Circulant embedding can be used to generate realizations of stationary normally distributed random fields provided that the sample points are uniformly spaced. As the associated covariance matrix C is BTTB and it can be extended to a larger $M \times M$ BCCB matrix B, samples of $N(\mathbf{0}, C)$ can be obtained from samples of $N(\mathbf{0}, B)$ by discarding some elements of the vector. Samples from $N(\mathbf{0}, B)$ can be obtained from the real and imaginary parts of $Y \sim CN(\mathbf{0}, 2B)$ using discrete Fourier transforms, as stated above.

The above should explain the fundamentals of the method used for generating the data a of problem (2.1) while working with the numerical example.

4.2 Monte Carlo method

In the project we do approximate expectations and this is done using the Monte Carlo method. This section discusses the principles of the Monte Carlo method. For details on the Monte Carlo method, see [8].

We consider a random variable X and the numerical evaluation of $\mathbb{E}[X]$ by the

empirical mean

$$\bar{X}_M := \sum_{m=1}^{M} X_m,$$

where $(X_m)_{m\geq 1}$ are independent simulations having the same distribution as X. Firstly we present the *strong law of large numbers*, which assures the almost sure convergence of $\bar{X}_M$ to X.

Theorem 4.2.1. *Let X be a real-valued integrable random variable. Then, with probability* 1*,*

$$\lim_{M\to\infty} \bar{X}_M = \mathbb{E}[X].$$

For a proof, see e.g. [8]. Let's continue with the definition of convergence in distribution.

Definition 4.2.1. A sequence of random variables $\bar{X}_M$ is said to *converge in distribution* to a random variable X as $M \to \infty$, denoted here by $\bar{X}_M \implies X$ as $M \to \infty$, if for any bounded continous function φ with compact support

$$\mathbb{E}[\varphi(X_M)] \to \mathbb{E}[\varphi(X)], \qquad \text{as } M \to \infty.$$

With this definition in mind, we continue with the important *central limit theorem.*

Theorem 4.2.2. *Let X be a d-dimensional square integrable random vector with covariance matrix C with entries $c_{ij} = \mathbb{C}\,\mathrm{ov}(X_i, X_j)$. Then*

$$\sqrt{M}(\bar{X}_M - \mathbb{E}[X]) \implies Z, \qquad \textit{as } M \to \infty$$

where $Z \sim N(0, C)$.

This means that if we want to increase the accuracy of $\mathbb{E}[X]$ by a factor of 10, it is necessary to increase M by a factor of $10^2 = 100$. Also, we note that the error is random and we can't have a control of the error with probability 1.

5
Algorithms

This chapter is the main chapter of the report as the algorithms used for computing the failure probabilities are described here. We start by describing the selective refinement algorithm which was used together with a multi-level Monte-Carlo method in [1] and [2]. Then a variation of that algorithm, including methods of how to choose sample sizes based on the Chapter 3 and 4 is described.

5.1 Selective refinement algorithm

Here we consider a model problem $\mathcal{M}$, e.g. a differential operator as in (2.1) and let u denote the unique solution to

$$\mathcal{M}(u, \omega) = 0, \tag{5.1}$$

where the data ω is sampled from a space Ω. We also consider a functional X of the solution u to (5.1). For this section, we follow the work of [1]. We consider refinement levels $\ell \in \mathbb{N}_0$ and let X'_ℓ be an approximation of X on level ℓ and $Q'_\ell := \mathbb{1}_{\{X'_\ell \leq y\}}$ be an approximation of Q on level ℓ. We will here approximate failure probability, also this on level ℓ. One first way of defining accuracy is by assuming

$$|X - X'_\ell| \leq \gamma^\ell, \tag{5.2}$$

for some $0 < \gamma < 1$, that is, all approximations on level ℓ are uniformly bounded by γ^ℓ. As we are interested in computing failure probabilities this assumption may be stronger than we need, since the failure probability functional Q is very sensitive to perturbations close to y but insensitive for perturbations far from y. This insensitivity is exploited in the next way of defining accuracy and in the upcoming algorithm.

Now, let X_ℓ be another approximation of X on level ℓ and $Q_\ell := \mathbb{1}_{\{X_\ell \leq y\}}$ be a corresponding approximation of Q on level ℓ. Here we instead define the following weakened condition

$$|X - X_\ell| \leq \gamma^\ell \qquad \text{or} \qquad |X - X_\ell| < |X_\ell - y|, \tag{5.3}$$

where $0 < \gamma < 1$ is fixed, called *tolerance factor*, and $y \in \mathbb{R}$ is the *critical value* y for which the failure probability should be computed. The selective refinement algorithm will be for computing an approximation X_ℓ fulfilling (5.3). In Figure 5.1 we see an illustration of how condition (5.3) looks like. The numerical error is allowed to be higher for values far from y.

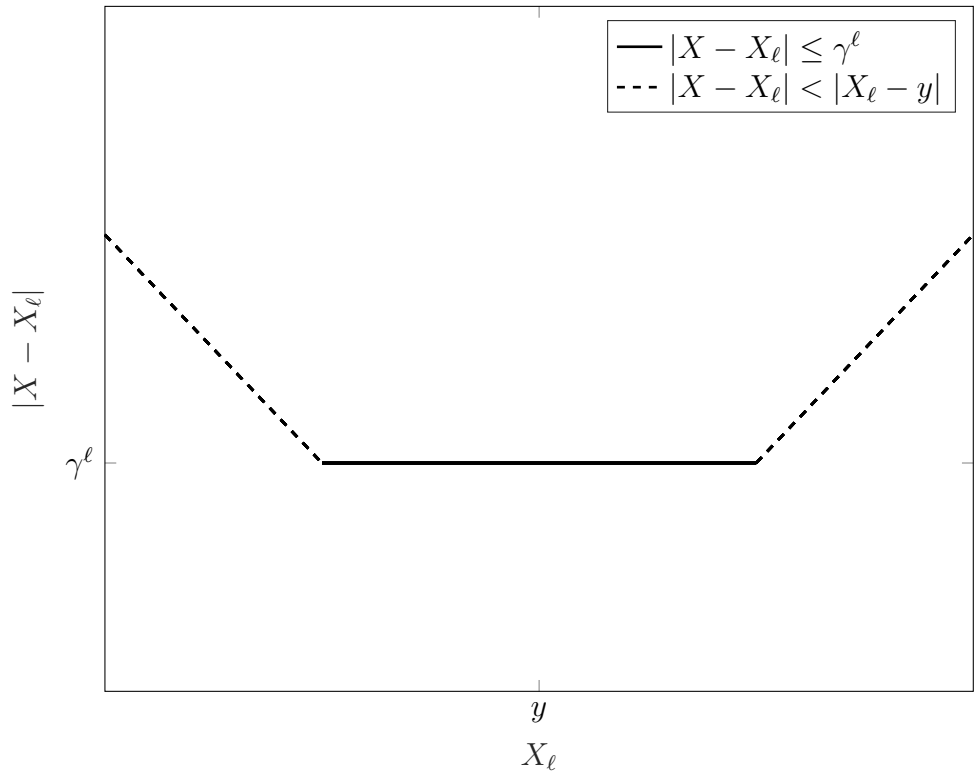


Figure 5.1: Illustration of (5.3)

The only regularity assumption on the model is that the cumulative distribution function (cdf), denoted by F, of the random variable $X(u)$ is Lipschitz continuous, i.e., there exists a constant C_L such that for any $x_1, x_2 \in \mathbb{R}$ it holds

$$|F(x_1) - F(x_2)| \leq C_L |x_1 - x_2|. \tag{5.4}$$

Next, as we have the approximation X_ℓ, we define $Q_\ell = \mathbb{1}_{\{X_\ell \leq y\}}$ analogously to Q. We are ready for the first lemma of the section.

Lemma 5.1.1. *Under assumptions (5.3) and (5.4), there exists a constant C, not depending on ℓ, such that*

$$|\mathbb{E}[Q_\ell - Q]| \leq C\gamma^\ell. \tag{5.5}$$

Proof. We split Ω into the events $A = \{\omega \in \Omega : \ \gamma^\ell \geq |X_\ell - y|\}$ and its complement $\Omega \setminus A$. For $\omega \in \Omega \setminus A$ we have, by (5.3), $|X - X_\ell| < |X_\ell - y|$. We then get $Q_\ell = Q$, since, if $X_\ell \leq y$,

$$\begin{aligned} |X_\ell - y| &= y - X_\ell \\ &\leq y - X + |X - X_\ell| \\ &\leq y - X + |X_\ell - y|, \end{aligned}$$

which means $X \leq y$, and, if $X_\ell > y$

$$\begin{aligned} |X_\ell - y| &= X_\ell - y \\ &\leq X - y + |X_\ell - X| \\ &< X - y + |X_\ell - y|, \end{aligned}$$

which means $X > y$. We have now seen, just using the triangle inequality, that for $\omega \in \Omega \setminus A$ we have $X_\ell \leq y \Leftrightarrow X \leq y$, which is $Q_\ell = Q$.

We also note that for the event A we have $|X_\ell - X| \leq \gamma^\ell$ by (5.3). This, in turn, implies $|X - y| \leq |X - X_\ell| + |X_\ell - y| \leq 2\gamma^\ell$ for the event A. Now we get,

$$\begin{aligned} |\mathbb{E}[Q_\ell - Q]| &= \left| \int_A Q_\ell(\omega) - Q(\omega) \ \mathrm{d}\mathbb{P}(\omega) \right| \\ &\leq \left| \int_A 1 \ \mathrm{d}\mathbb{P}(\omega) \right| \\ &\leq \mathbb{P}(|X - y| \leq 2\gamma^\ell) \\ &= F(y + 2\gamma^\ell) - F(y - 2\gamma^\ell) \\ &\leq 4C_L\gamma^\ell, \end{aligned}$$

using that $Q_\ell = Q$ for $\omega \in \Omega \setminus A$ in the first line, and the Lipschitz continuity assumption (5.4) in the last line. □

A main feature of the selective refinement algorithm is that it exploits the fact that $Q_\ell = Q$ for realizations with $|X - X_\ell| < |X_\ell - y|$. This means that even if the error is greater than γ^ℓ, it might be suffciently accurate to yield the correct value of Q_ℓ. The algorithm is presented below:

Algorithm 1 Selective refinement algorithm

1: Input arguments: level ℓ, realization i, critical value y, tolerance factor γ.
2: Compute $X_0'(\omega_\ell^i)$.
3: Let $j = 0$.
4: **while** $j < \ell$ and $\gamma^j \geq |X_j'(\omega_\ell^i) - y|$
5: Let $j = j + 1$.
6: Compute $X_j'(\omega_\ell^i)$.
7: **end while**
8: Let $X_\ell(\omega_\ell^i) = X_j'(\omega_\ell^i)$

Lemma 5.1.2. *Approximations X_ℓ computed using Algorithm 1 satisfy (5.3).*

Proof. In each realization in the while-loop of Algorithm 1 we have that γ^j is the error tolerance, i.e., $|X(\omega_\ell^i) - X_j'(\omega_\ell^i)| \leq \gamma^j$. The stopping criterion hence implies (5.3). □

Now, in [1], Algorithm 1 is used together with a multi-level Monte-Carlo method. The selective refinement algorithm is based on error estimating. What we do instead is using a new variation of the selective refinement algorithm described in the next section.

5.2 Variation of the selective refinement algorithm

This variation of the selective refinement algorithm is performed for a demonstrational example and the main example, the PDE problem (2.1), and aims for finding an efficient way of computing the failure probability for that problem. We put focus on sample sizes and mesh discretizations according to the error analysis of Chapter 3. The selective refinement algorithm in [1] deals with one realization and individual error estimates and then uses a multilevel Monte-Carlo method for computing an approximator of the failure probability (it is not necessary to use a multilevel Monte Carlo method together with the selective refinement algorithm). Now we instead present an algorithm for directly, by means of a standard Monte Carlo method, compute an approximation of the failure probability. This is to be compared against the selective refinement algorithm with a standard Monte-Carlo method. In the previous work of [1] a posteriori error estimates was used which demanded for a deeper refinement for all samples to compute errors. Here, we instead iterate until we get a certain convergence in p so that we do not need error estimates for each sample.

Also here, we introduce a hierarchy of levels $\ell = 0, 1, \ldots, L$, which correspond to different meshes of D. We let N be a start sample size (a sample size on level $\ell = 0$). We also assume we have a given functional X of the solution, which we are able to compute directly as we have the solution. An example of such an X is $X(u) = \max_{x \in D} u(\cdot, x)$, which will be used later on in a numerical experiment. The first thing to do in the algorithm is generating N random fields on the finest mesh discretization (which correspond to level L). We want to clarify that we only generate random fields on the finest discretization. We assume that the nodes of coarser discretizations are subsets of the nodes of the finer discretizations, i.e., $\mathcal{N}_0 \subset \mathcal{N}_1 \subset \cdots \subset \mathcal{N}_L$, if we denote by $\mathcal{N}_\ell$ the set of nodes for the discretization on level ℓ. For the computations on coarser levels than level L, we use the values for the relevant nodes.

As we have N generated random fields, the next step is to decide sample sizes N_ℓ for $\ell = 0, 1, \ldots, L$. We want to point out that we are not taking new samples for each level. The sample sizes are for deciding the amount of the samples which should be refined. The idea is to have N_0 samples, make computations for them on level $\ell = 0$, then choose N_1 of them with solutions close to y and make computations for them on level $\ell = 1$. In this fashion we proceed until level $\ell = L$.

Here we assume

$$\begin{aligned} N_0 &= N \\ N_1 &= C\delta N \\ N_2 &= C\delta^2 N \\ &\vdots \\ N_L &= C\delta^L N, \end{aligned} \tag{5.6}$$

for some fix $0 < C < 1$, $0 < \delta < 1$. This means, we are refining a factor of $C\delta^\ell$ of the start sample size on level ℓ. This assumption is based on Lemma 5.1.1. For a PDE discretization as we work with in this project, δ depends on h. For example, if the convergence is h^2 then $\delta = \frac{1}{4}$.

We want to find a good way of deciding C and δ. First we decide δ as in Algorithm 2. We let δ be the average, with respect to the levels, of the average change of the functional of the solution.

Algorithm 2 Algorithm for finding δ

1: Initialize sample size N_δ ($\ll N$)
2: Generate realizations $\{\omega_i\}_{i=1}^{N_\delta}$ on the finest level L
3: **for** $i = 1, \dots, N_\delta$
4: **for** $\ell = 0, \dots, L$
5: Solve (3.5) with A given by the random field generated by ω_i and where V_h depend on ℓ, to get $u_\ell(\omega_i)$
6: Compute $X(u_\ell(\omega_i))$
7: Let $\delta = \frac{1}{L}\sum_{\ell=0}^{L-1} \frac{1}{N_\delta}\sum_{i=1}^{N_\delta} \frac{X(u_{\ell+1}(\omega_i))}{X(u_\ell(\omega_i))}$

By "Solve (3.5) with A given by the random field generated by ω_i" we mean that A represent the random field which becomes given as the realization is done, via circulant embedding. The dependency of V_h on ℓ is such that the discretization is finer for deeper levels ℓ.

To then find C and the approximate failure probability $\tilde{p}$, we use an iterative algorithm where we assume that δ is already given by Algorithm 2. We assume that we have an exact solution $u(\omega, \cdot)$ as we have a realization ω. In practice, we get this solution by solving the finite element problem for a small h; a smaller h than for $\ell = L$.

The idea is to first choose C_0 close to 0 and get sample sizes according to (5.6) with $C = C_0$. Since C_0 is close to 0 the sample sizes N_ℓ will decrease fast as ℓ increases. As we performed Algorithm 2 we had a fix sample size N_δ (same for each level ℓ) but now we do not have such a sample size. Here we instead, while going from level ℓ to $\ell+1$, choose the $N_{\ell+1}$ realizations which have resulted in a QoI $X(u)$ closest to y. This is because those are most likely to "change side" (by that we mean with respect to y if we think of a real line with y in the middle). This is illustrated in Figure (5.2). At the end, it will mean that for some realizations, the QoI will only be computed at level $\ell = 0$, for some realizations, the QoI will be computed at level $\ell = 1$ and some realizations (N_L realizations) will be computed at the finest accuracy (level $\ell = L$). The failure probability will in the end be computed as

$$\tilde{p}_0 = \frac{1}{N_0}\sum_{i=1}^{N_0} \mathbb{1}_{\{X(u(\omega_i))\leq y\}},$$

that is, by means of a Monte-Carlo method (described in Chapter 4).

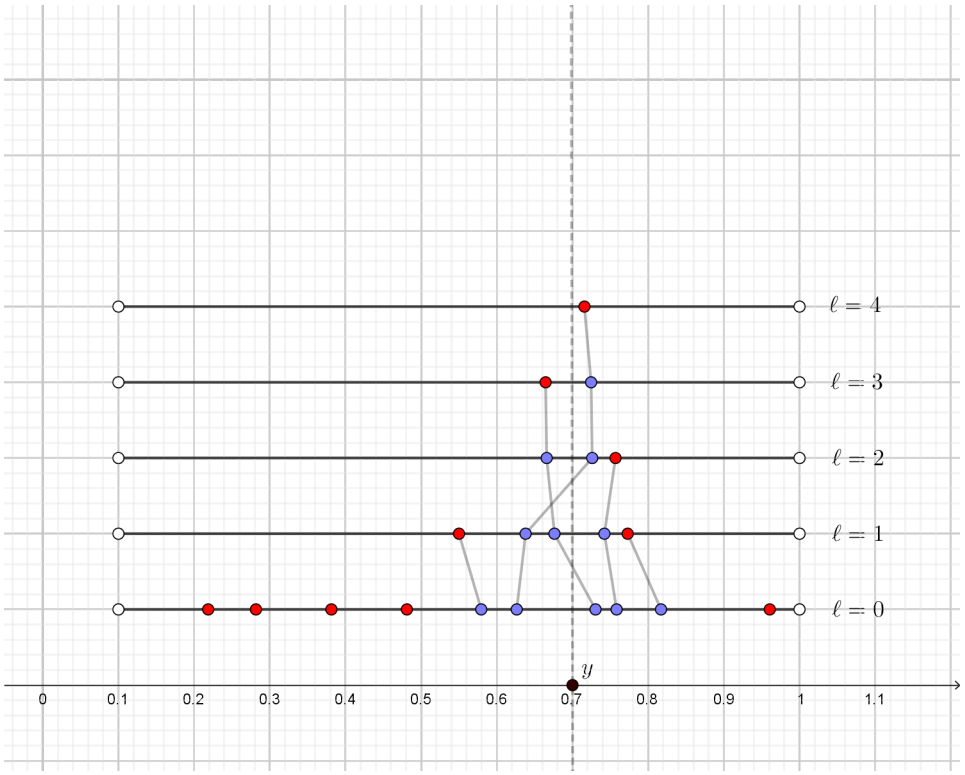


Figure 5.2: An illustration of one iteration of Algorithm 3 with $L = 4$, $N_0 = 10$, $N_1 = 5$, $N_2 = 3$, $N_3 = 2$, $N_4 = 1$. The blue dots are to be refined and the red dots are finished refined.

As we have also chosen $C_1 > C_0$ and performed the same thing as for $\tilde{p}_0$, but with new sample sizes according to (5.6) with $C = C_1$, to get $\tilde{p}_1$ we are able to compare the outcomes. We want the derivative of $\tilde{p}$ with respect to C to be low. In practice we decide a tolerance *tol* and look at $\left|\frac{\tilde{p}_1 - \tilde{p}_0}{C_1 - C_0}\right|$. A stopping criterion for the algorithm will be

$$\left|\frac{\tilde{p}_{k+1} - \tilde{p}_k}{C_{k+1} - C_k}\right| < tol. \tag{5.7}$$

Algorithm 3 Iterative algorithm for finding C and $\tilde{p}$

1: Initialize sample size N, tolerance tol, C_0, $k = 0$
2: Generate realizations $\{\omega_i\}_{i=1}^{N}$ on the finest level L
3: Decide sample sizes $\{N_\ell\}_{\ell=0}^{L}$ according to (5.6) with $C = C_0$
4: **for** $i = 1, \ldots, N$
5: Solve (3.5) with A given by the random field generated by ω_i and with the coarsest mesh (corresponding to $\ell = 0$), to get $u_0(\omega_i)$
6: Compute $X(u_0(\omega_i))$
7: **for** $\ell = 1, \ldots, L$
8: Choose the N_ℓ of the realizations $\{\omega_i\}_{i=1}^{N_{\ell-1}}$ with $X(u_{\ell-1}(\omega_i))$ closest to y
9: **for** the chosen realizations $\{\omega_j\} \subset \{\omega_i\}_{i=1}^{N_{\ell-1}}$
10: Solve (3.5) with A given by the random field generated by ω_i, where V_h depend on ℓ, to get $u_\ell(\omega_j)$
11: Compute $X(u_\ell(\omega_j))$
12: Let $\tilde{p}_k = \frac{1}{N_0} \sum_{i=1}^{N_0} \mathbb{1}_{\{X(u_*(\omega_i)) \leq y\}}$, where level index $*$ denotes the deepest level $u(\omega_i)$ have been solved for
13: Choose $C_1 > C_0$ and repeat Step 3-12 with $C = C_1$ in Step 3 to get $\tilde{p}_1$
14: **while** (5.7) is not fulfilled and $C < 1$
15: $k = k + 1$
16: Choose $C_{k+1} > C_k$ and repeat Step 3-12 with $C = C_{k+1}$ in Step 3 to get $\tilde{p}_{k+1}$
17: Let $C = C_k$
18: Let $\tilde{p} = \tilde{p}_k$

The choices of C_k for Step 13 and 16 have in this project been made with $C_0 = 0.1$ and $C_{k+1} = C_k + 0.1$, but other, might be non-linear, growths of C_k are also possible. The reasons of choosing C_k's in the way it is done are simplicity and that it worked well for numerical examples.

6
Results

This chapter will contain the numerical experiments done in this project. The two examples is of different types, where the first is of demonstrational purpose while the second is testing the method (Algorithm 2 and 3) for a quantity of interest of the solution of (2.1). Lastly, we discuss the results and future work in the third section,

6.1 A distribution example

In this numerical experiment we are not considering a PDE. Here instead, inspired by [1], we consider the standard normal distribution. We let the QoI X belong to the standard normal distribution. Also, we let $y = 0.5$ and we approximate the failure probability $\mathbb{P}(X \leq y)$. The true value of this probability is $\Phi(0.5) \approx 0.69146$. The implementation of this example was done using MATLAB.

For realizations ω of the standard normal distribution we let the approximations of X be given by $X_h(\omega) = \omega + h\frac{2U(\omega,h)-0.9}{1.1}$, where $U(\omega, h)$ is a uniformly distributed random number between 0 and 1. The parameters ω and h of U are to clarify that it is a specific random number for each ω and h. We let $\ell = 0$ correspond to $h = 0.1$, $\ell = 1$ correspond to $h = 0.01$, $\ell = 2$ correspond to $h = 0.001$, $\ell = 3$ correspond to $h = 0.0001$ and $\ell = 4 =: L$ correspond to $h = 0.00001$. We clarify that for the algorithms this will mean that Step 5 of Algorithm 2 will not be needed and Step 6 of Algorithm 2 should be interpreted as computing $X_h(\omega_i)$ (and similarly for Step 5 and 6 and Step 10 and 11 of Algorithm 3). We used sample sizes $N_\delta = 100$ and $N = 10^6$ for Algorithm 2 and 3 respevtively.

As we performed the algorithms we ended up with $\delta \approx 0.32042$ and $C = 0.3$. This means that the amount of calculations were $N_0 = N = 10^6$ for level $\ell = 0$, $N_1 = C\delta N = 96126$ for level $\ell = 1$, $N_2 = C\delta^2 N = 30801$ for level $\ell = 2$, $N_3 = C\delta^3 N = 9870$ for level $\ell = 3$ and $N_4 = C\delta^4 N = 3163$ for level $\ell = L = 4$. Finally, $\tilde{p} \approx 0.692144$, if we denote by $\tilde{p}$ the derived approximation of $\mathbb{P}(X \leq y)$. This is to be compared with the Monte Carlo result $p_{MC} = 0.692142$, which is computed as

$$p_{MC} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{\omega_i \leq y},$$

if we denote by $\{\omega_i\}_{i=1}^N$ the sample drawn from the standard normal distribution.

Assume now that the computational cost $\mathcal{C}_\ell$ for one calculation on level ℓ is

$$\mathcal{C}_\ell = 4^\ell, \tag{6.1}$$

which is realistic if working with e.g. PDEs and the finite element method where levels ℓ correspond to different refinements of the finite element mesh. Then the cost for the last iteration of Algorithm 4, which computes $\tilde{p}$, is

$$\begin{aligned} \text{Cost} &= \sum_{\ell=0}^{4} N_\ell 4^\ell \\ &\approx 3.32 \times 10^6, \end{aligned}$$

which is to be compared against the cost of performing the Monte Carlo method, using the same cost model (6.1) for the deepest level ℓ. This total computational cost becomes

$$\begin{aligned} \text{Cost} &= N \times 4^L \\ &\approx 2.56 \times 10^8. \end{aligned}$$

6.2 The maximum

In this section we will describe the results of our algorithm described in Section 5.2 implemented for problem (2.1). The implementation was done using Python coding together with the FEniCS software, see [6]. Throughout the chapter, we consider $f \equiv 1$ and $D = [0,1] \times [0,1]$ in (2.1).

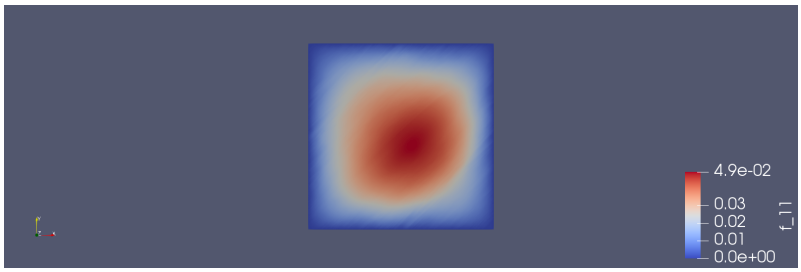


Figure 6.1: Solution u for problem (2.1)

In Figure 6.1, which was created using ParaView (see [9]), we see how the solution u behaves for our problem for one realization of a, with $\sigma = 1$ and $\rho = 0.1$. We continue with a numerical experiment where Algorithm 2 and 3 are tested.

In this numerical experiment we let the QoI X be given by

$$X(u) = \max_{x \in D} u(x),$$

and we seek an approximation $\tilde{p}$ of $p = \mathbb{P}(\{X \leq y\})$. We let $\ell = 0$ correspond to a triangulation of D with $2^4 \times 2^4$ nodes, $\ell = 1$ correspond to a triangulation of D

with $2^5 \times 2^5$ nodes, $\ell = 2$ correspond to a triangulation of D with $2^6 \times 2^6$ nodes and finally $\ell = 3 =: L$ correspond to a triangulation of D with $2^7 \times 2^7$ nodes. These nested meshes of D for $\ell = 0$ and $\ell = 1$ could be seen in Figure 6.2.

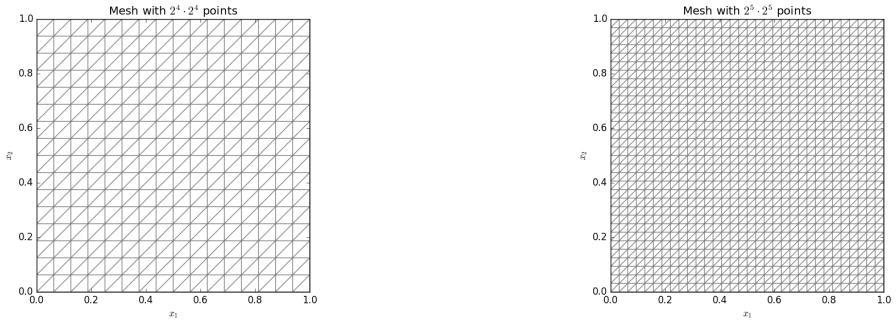


Figure 6.2: The triangulations of D used for level $\ell = 0$ and $\ell = 1$

The parameters used for the log-normal field was here $\rho = 0.1$ and $\sigma = 0.46$, inspired by [2]. We initialized sample sizes $N_\delta = 50$ for Algorithm 2 and $N = 1000$ for Algorithm 3. Also, we chose $tol = 0.1$ and $y = 0.075$; the choice of y just to not get a probability close to 1 or 0. With these parameters we got $\delta \approx 0.49383$, $C = 0.5$ and $\tilde{p} = 0.611$. As we performed the Monte Carlo method with the finest discretization (the triangulation of D with $2^7 \times 2^7$ nodes) we obtained $p_{\text{MC}} = 0.6$. Hence we got the error

$$|\tilde{p} - p_{\text{MC}}| = 0.011.$$

The computational cost of performing the last iteration of Algorithm 3 is

$$\begin{aligned} \text{Cost} &= \sum_{\ell=4}^{7} N_\ell 4^\ell && (6.2) \\ &\approx 1.56 \times 10^6, && (6.3) \end{aligned}$$

which is to be compared with the computational cost of performing the Monte Carlo method for the finest discretization. This cost is

$$\begin{aligned} \text{Cost} &= N \times 4^{L+3} \\ &\approx 1.64 \times 10^7. \end{aligned}$$

We notice how the cost is approximately 10 times lower with our algorithm than it is for the Monte Carlo method in this case. In the calculations of the computational cost the factor 4 is involved because of the number of points in the meshes used. Each level ℓ is associated to a mesh with $2^6 \times 4^\ell$ points.

6.3 Discussion

The conclusion for the results is that the example with the maximum as functional (see section 6.2) worked out well. The algorithm performed well for that problem.

Worth noticing is that we did only take into account the *last* iteration of Algorithm 3 and not the total cost of the method (the full Algorithm 2 and 3) while comparing costs with the Monte Carlo method. This was because of, firstly, that the cost of Algorithm 2 is negligible due to the low sample size and, secondly, that the last iteration of Algorithm 3 is the most computationally heavy of that algorithm. If we instead look at all iterations of Algorithm 3 and compute the computational cost for the second numerical example the result becomes

$$\begin{aligned}\text{Cost} &= \sum_{C\in\{0.1,0.2,\ldots,0.5\}} \sum_{\ell=4}^{7} N_{C,\ell} 4^{\ell} \\ &\approx 6.58 \times 10^{6},\end{aligned}$$

if we denote by $N_{C,\ell}$ the sample size corresponding to C and the level ℓ, and this is still lower than the computational cost of performing the Monte Carlo method for the finest discretization, see (6.2).

Further, we have in this project only looked at numerical examples and not dealt with theoretical convergence analysis of the approximations to the real solution. We suggest it as future work. For future work we also suggest testing other QoIs than the maximum. Here, we looked at the QoI

$$X(u) = \max_{x\in D} u(x),$$

as it was fairly straight-forward to implement but of course other QoIs, such as

$$X(u) = \int_D u(x)\ \mathrm{d}x,$$

could be of interest.

One more suggestion of future work could be to analyze the role of N_δ and N and how the choices of these parameters could be optimized. For us, they have been chosen experimentally.

Bibliography

[1] D. Elfverson, F. Hellman, and A. Målqvist. (2016). A multilevel Monte Carlo method for computing failure probabilities. SIAM/ASA Journal on Uncertainty Quantification, 4, 312–330. https://doi.org/10.1137/140984294

[2] F. Fagerlund, F. Hellman, A. Målqvist and A. Niemi. (2016). Multilevel Monte Carlo methods for computing failure probability of porous media flow systems. Advances in Water Resources, 94, 498–509. https://doi.org/10.1016/j.advwatres.2016.06.007

[3] J. Charrier. (2012). Strong and weak error estimates for elliptic partial differential equations with random coefficients, SIAM J. Numer. Anal., 50, 216–246

[4] C.E. Powell, School of Mathematics. Generating Realisations of Stationary Gaussian Random Fields by Circulant Embedding.

[5] D. Braess, Finite elements: theory, fast solvers, and applications in solid mechanics, Cambridge University Press, 2007.

[6] A. Logg, K-A. Mardal and G. Wells. (2012). Automated Solution of Differential Equations by the Finite Element Method, Lect. Notes Comput. Sci. Eng. 84, Springer, Berlin, Heidelberg

[7] C. Dietrich and G. Newsam. Fast and exact simulation of stationary gaussian processes through circulant embedding of the covariance matrix. SIAM J. Sci. Comput., 18(4):1088–1107, 1997.

[8] E. Gobet (2016) Monte-Carlo Methods and Stochastic Processes: From Linear to Non-Linear. (Chapman Hall/CRC, Boca Raton, FL).

[9] U. Ayachit. (2015). The ParaView Guide: A Parallel Visualization Application, Kitware, ISBN 978-1930934306

[10] S. Brenner and R. Scott. (2008). The Mathematical Theory of Finite Element Methods. Springer-Verlag New York.