Politecnico di Torino
Facoltà di Ingegneria I
Corso di Laurea in Ingegneria Matematica

King Abdullah University of Science and Technology
Center for Uncertainty Quantification in Computational Science and Engineering

Master Thesis

Multilevel Monte Carlo method for PDEs with fluid dynamic applications

Advisor:
Prof. Claudio CANUTO
Co-advisor:
Dr. Matteo ICARDI
Co-advisor:
Prof. Raúl TEMPONE

Candidate:
Nathan QUADRIO

October 2014

Prof. *Claudio Canuto*
Department of Mathematics,
Polytechnic University of Turin,
Turin, Italy.

Dr. *Matteo Icardi*
Division of Mathematics & Computer, Electrical and Mathematical Sciences & Engineering
King Abdullah University of Science and Technology,
Thuwal, Kingdom of Saudi Arabia.

Prof. *Raúl Tempone*
Division of Mathematics & Computer, Electrical and Mathematical Sciences & Engineering
King Abdullah University of Science and Technology,
Thuwal, Kingdom of Saudi Arabia.

To Anita,
a light for me when all the other lights went out.

“In mathematics you don’t understand things. You just get used to them.”

John von Neumann

Abstract

This thesis focuses on PDEs in which some of the parameters are not known exactly but affected by a certain amount of uncertainty, and hence described in terms of random variables/random fields. This situation is quite common in engineering applications.

A common goal in this framework is to compute statistical indices, like mean or variance, for some quantities of interest related to the solution of the equation at hand ("uncertainty quantification"). The main challenge in this task is represented by the fact that in many applications tens/hundreds of random variables may be necessary to obtain an accurate representation of the solution variability. The numerical schemes adopted to perform the uncertainty quantification should then be designed to reduce the degradation of their performance whenever the number of parameters increases, a phenomenon known as "curse of dimensionality".

A method that acts in this direction is Monte Carlo sampling. Such method is known to be dimension independent and very robust but it is also known for is very slow convergence rate. In this work we describe Monte Carlo sampling together with a solid error analysis and we provide a test for its robustness by integrating different functions with different regularitues and by solving different PDE problems with random coefficients.

Later on we introduce the technique of variance reduction and a further application of this idea that goes with the name of Multilevel Monte Carlo (MLMC) method. The asymptotic cost of solving the stochastic problem with the multilevel method is proved to be significantly lower than that of the standard method and, in certain circumstances, grows only proportionally with respect to the cost of solving the deterministic problem. Numerical calculations demonstrating its effectiveness are presented and more complex problems such as elliptic PDEs in a domain with random geometry are also presented, this last task has been performed to test a code designed for the simulation of flow through porous media at the pore scale. The results are promising considering the very complex geometries that require extremely expensive discretizations.

This work is the final outcome of the participation to the Visiting Student Research Program at the King Abdullah University of Science and Technology, a project that allows students

to conduct research with faculty mentors in selected areas of pure and applied sciences, and a Visiting Research Fellowship at the University of Texas in Austin.

Keywords: Uncertainty Quantification, Monte Carlo Sampling, Variance Reduction, Multi-level Monte Carlo.

Acknowledgements

This thesis is the result of the fruitful collaboration among many people, who deserve grateful acknowledgement.

My deepest thanks goes to Matteo Icardi for the huge support in every aspect of this work, from the inspiring discussions at the blackboard to the technical guidance and suggestions in the numerical results, to the proof-reading of every single page and slide I have written, with 24/7 availability.

A huge thanks goes also to Raúl Tempone: in addition to the warm hospitality in the many places I have visited him and the contagious enthusiasm in facing any mathematical challenge, I always have greatly benefitted from his help in discussing and dissecting any problem we had to solve and from his many suggestions.

I am also greatly thankful to Claudio Canuto, who is and will always be a great inspiration for me an my work, for the overall help and the infinite patience.

At KAUST, I would like to thank David Yeh, who made for me the whole KAUST experience possible and great.

This thesis would not have been possible without the amazing and constant support I got from my beloved ones: my parents, my grandma and my sister. The grand finale is however for Anita, who is my love, my strength, my hope and my change of perspective. This thesis is dedicated to her.

Turin, October 2014 Nathan Quadrio

Contents

Abstract **5**

Acknowledgements **6**

1 Introduction to Uncertainty Quantification **12**

1.1 Forward uncertainty propagation 14

1.2 Inverse problem within uncertainty quantification 16

1.3 Overview 19

2 Monte Carlo Method **20**

2.1 Introduction 20

2.2 Monte Carlo method 21

2.2.1 Numerical tests 25

2.3 Quasi-Monte Carlo method 28

2.3.1 Error analysis in 1D 30

2.3.2 Numerical tests 32

2.4 Monte Carlo method for PDEs with random coefficient 34

2.4.1 Error analysis 34

2.4.2 Complexity analysis 38

2.4.3 Numerical tests 39

2.5 Variance Reduction 44

2.5.1 Control Variate 44

3 Multilevel Monte Carlo Method **47**
3.1 Introduction . . . 47
3.2 Multilevel Monte Carlo method . . . 48
3.2.1 MLMC implementation . . . 51
3.3 Numerical tests . . . 53
3.4 Application to random geometry problems . . . 56
3.4.1 Heterogeneous materials . . . 56
3.4.2 The geometry generation . . . 58
3.5 Simulations . . . 61
3.5.1 Elliptic PDE with random forcing . . . 61
3.5.2 Elliptic PDE with a single random inclusion . . . 63
3.5.3 Diffusion on a randomly perforated domain . . . 66
3.5.4 Pore-scale Navier-Stokes . . . 68
3.6 Conclusions . . . 72

APPENDIX **72**

A Multilevel Monte Carlo Pore-Scale Code **73**
A.1 Main purpose . . . 74
A.2 Structure . . . 74
A.3 Final statements . . . 76

List of Figures

1.1 Forward uncertainty propagation. 15
1.2 Bayesian inverse problem. 16
1.3 Uncertainty quantification in bayesian inversion. 18

2.1 Deterministic quadrature (left) vs. Monte Carlo sampled (right) points in the case $d = 2$. 23
2.2 1D functions used to test the MC integration (left column) and convergence of the computational error as $1/\sqrt{M}$ (right column). 27
2.3 Idea of discrepancy (left) and example of Sobol sequence (right) in a two-dimensional domain. The discrepancy can be also seen more intuitively as $\Delta_P = \frac{\#\text{ points in } [\mathbf{0},\mathbf{x}]}{M} -$ Vol$([\mathbf{0},\mathbf{x}])$. 29
2.4 1D functions used to test the MC integration (left column) and convergence of the computational error (right column) as $1/\sqrt{M}$ for the MC and $1/M$ for the "good cases" of QMC. 33
2.5 Numerical simulation for Model 1 - Mesh refinement (left column, $I = 1/h = 64, 128, 512$) and increase of the variability (right column $N = 128, 256, 512$). The phenomenon of *homogenization* can be seen very well in the second column. . . 42
2.6 Numerical simulation for Model 2 - Different numbers of random variables (left column $N = 1, 4, 8$) and different number of samples (right column $M = 10^2, 10^3, 10^4$). 43

3.1 Numerical result for the 1D elliptic PDE with random diffusion solved modeled with a piecewise constant coefficient. Rates of strong and weak error (up-left and up-right plots resp.), consistency check (Eq. 3.3) and kurtosis (middle-left and middle-right plots resp.), accuracy vs. cost (low-right) and sample vs. level for different accuracies (low-left). 55
3.2 Examples of random heterogeneous materials. Left panel: A colloidal system of hard spheres of two different sizes. Right panel: A Fontainebleau sandstone. Images from [37]. 57
3.3 Random geometry realizations. 59
3.4 Results for the QoI in the elliptic PDE with random forcing. 61
3.5 Results for the QoI in the elliptic PDE with random forcing - γ *estimate* 62
3.6 Results for the elliptic PDE with a single random inclusion. 63
3.7 Results for the elliptic PDE with a single random inclusion. 64
3.8 Results for the diffusion on a randomly perforated domain. 67
3.9 Mesh of a geometry realization. 69
3.10 Results for the incompressible Navier-Stokes flow simulation.. 70

A.1 Pore-Scale Code. 73

List of Tables

3.1 Numerical result for the 1D elliptic PDE with random diffusion solved modeled with a piecewise constant coefficient. 54

3.2 Cost comparison between MLMC and StdMC as $\varepsilon \to 0$ for the 1D elliptic PDE with random diffusion. 54

3.3 Different classes of steady-state effective media problems considered here. $F \propto K_e \dot{G}$, where K_e is the general effective property, G is the average (or applied) generalized gradient or intensity field, and F is the average generalized flux field. Class A and B problems share many common features and hence may be attacked using similar techniques. Class C and D problems are similarly related to one another. 58

3.4 Results for the elliptic PDE with random forcing. 63

3.5 Results for the diffusion in a random domain. 66

3.6 Results for the incompressible Navier-Stokes flow simulation. 71

3.7 Predicted cost of the MLMC estimator given by Theorem 4 to achieve a MSE of ε^2 compared to the cost of the standard MC estimator given by Equation 2.4. 72

CHAPTER 1

Introduction to Uncertainty Quantification

Since more and more powerful computer are being developed, allowing for more complex models to be solved, it is important to assess if the mathematical and the computational models are accurate enough and, in general, if one can establish an "error bar" on the results or a more accurate quantification of its variability. Uncertainty Quantification (UQ) aims at developing rigorous methods to characterize the impact of "limited knowledge" on model parameters of the quantities of interest. As UQ is at the interface of physics, mathematics and statistics, a deep understanding of the physical problem of interest is required as well as its mathematical description and a good probabilistic framework. In fact the most modern UQ approaches are a combination of numerical analysis and statistical techniques.

Even if numerical simulations have reached a wide spread use and success in terms of lowering production costs and of reduction of physical prototyping, it still remains difficult to provide a certain confidence level in all the information obtained from numerical predictions. This difficulty mainly comes from the amount of uncertainties related to the inputs of any computation attempting to represent a physical system. As a consequence, for some applications we still prefer the physical tests, so that the quantification of the *errors* and *uncertainties* becomes necessary in order to establish the numerical simulations predictive capabilities.

Classically, the errors leading to discrepancies between simulations and real world systems are grouped into three distinct families:

- **Model error:** Simulations rely on the resolution of mathematical models taking into ac-

count the main characteristics of the system being studied. Often, simplifications of the models are performed in order to facilitate their resolution and based on some assumptions with the direct consequence of modeling some sort of ideal system different form the real one. What one is expecting is that the predictions based on the simplified model will remain sufficiently accurate to conduct a suitable analysis.

- **Numerical error:** When a mathematical model is discretized, numerical errors are introduced since the numerical methods provide usually only an approximation of the exact model. These errors can be reduced to an arbitrarily low level by using finer discretizations and, therefore, more computational resources. To this aim, it is important to design numerical methods incorporating specific measures, based for instance on notions of *convergence*, *consistency* and *stability*. It is known that these errors will always be nonzero due to the finite representation of numbers in computers.

- **Data error:** Mathematical models need to be specified with parameters and data regarding for instance geometry, boundary and initial conditions and external forcings. While parameters may be physical or model constants, data cannot be exactly specified because of limitations in experimental data available, in the knowledge of the system or because of inherent variability of the system studied (e.g. the porosity in a porous medium). Using data which partially reflect the nature of the exact system induces additional errors, called data errors, to the prediction.

The latter are usually referred as uncertainties and a more precise characterization based on the distinction in aleatory and epistemic uncertainties can be done:

- **Aleatory uncertainty:** It is the physical variability present in the system being analyzed or its environment. It is not strictly due to lack of knowledge and cannot be reduced. Additional experimental characterization might provide more conclusive evidence of the variability but cannot eliminate it completely. Aleatory uncertainty is normally characterized using probabilistic approaches.

- **Epistemic uncertainty:** It is a potential deficiency that is due to a lack of knowledge. It can arises from assumption introduced in the derivation of the mathematical model used or simplifications related to the correlation or dependence between physical processes. It is possible to reduce the epistemic uncertainty by using, for example, a combination of calibration, inference from experimental observations and improvement of the physical models. It is not easily characterized by probabilistic approaches because it might be

difficult to infer any statistical information due to the nominal lack of knowledge. Typical examples of sources of epistemic uncertainties are turbulence model assumptions and surrogate chemical kinetics models.

To sum up, what uncertainty analysis aims to is identifying the overall output uncertainty in a given system.

There are two major types of problems in uncertainty quantification: one is the forward propagation of uncertainty and the other is the inverse assessment of model uncertainty and parameter uncertainty. There has been a proliferation of research on the former problem and a number of numerical analysis techniques were developed for it. On the other hand, the latter problem is drawing increasing attention in the engineering design community, since uncertainty quantification of a model and the subsequence predictions of the true system response(s) are of great interest in both robust design and engineering design making.

1.1 Forward uncertainty propagation

Once a suitable mathematical model of a physical system is formulated, the numerically simulation task typically involves several steps.

Initially one has to specify the case, i.e. the input parameters. Generally, one needs to state exactly the geometry associated with the system and, in particular, the computational domain. Boundary conditions have also to be imposed. In the case of transient systems, initial conditions are also provided and, when present, external forcing functions are applied to the system. In the end physical constants are also specified in order to describe the properties of the system, as well as modeling or calibration data. The next step is the simulation. One has to define a computational grid on which the model solution is first discretized. Additional parameters related to time integration, whenever relevant, are also specified. Numerical solution of the resulting discrete analogue of the mathematical model can then be performed. Here one should choose a deterministic numerical model having a well-posed mathematical formulation in the sense of Hadamard, i.e. one wants that the mathematical model admits the existence of a unique solution with continuos dependence from the data, and with which can be achieved small discretization errors. The final step concerns the analysis of the computed solution.

The simulation methodology above reflects an idealized situation that may not be always achieved in practice. In fact, in many cases, the input data set may not be completely known due to the reasons mentioned before. Thus, though model equations may be deterministic,

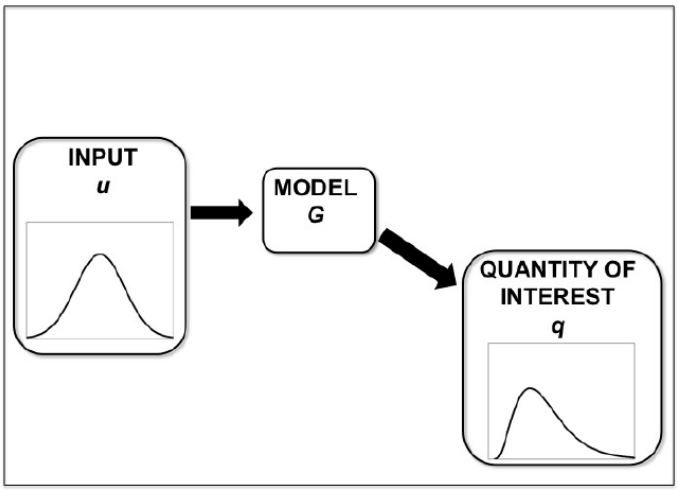


Figure 1.1: Forward uncertainty propagation.

it may not be possible to rely on a single deterministic simulation. A probabilistic framework is useful to represent the variability of the input data, that provide a variance analysis which characterize a confidence measure in compute predictions and a risk analysis to determine the probabilities of the system exceeding certain critical values. Within a probabilistic framework, the problem of uncertainty propagation consists of the generation of PDFs of the outcomes given distribution of all the parameters.

Uncertainty propagation is the quantification of uncertainties in system output(s) propagated from uncertain inputs. It focuses on the influence of the parametric variability, listed in the sources of uncertainty on the outputs. The targets of uncertainty propagation analysis can be:

- To evaluate low-order moments of the outputs, i.e. mean and variance.
- To evaluate the reliability of the outputs. This is especially useful in reliability engineering where outputs of a system are usually closely related to the performance of the system.
- To assess the complete probability distribution of the outputs. This is useful in the scenario of utility optimization where the complete distribution is used to calculate the utility.

Existing uncertainty propagation approaches include probabilistic approaches and non-probabilistic approaches.

Now, let X, R be Banach spaces and $G : X \to R$. For example G might represent the *forward*

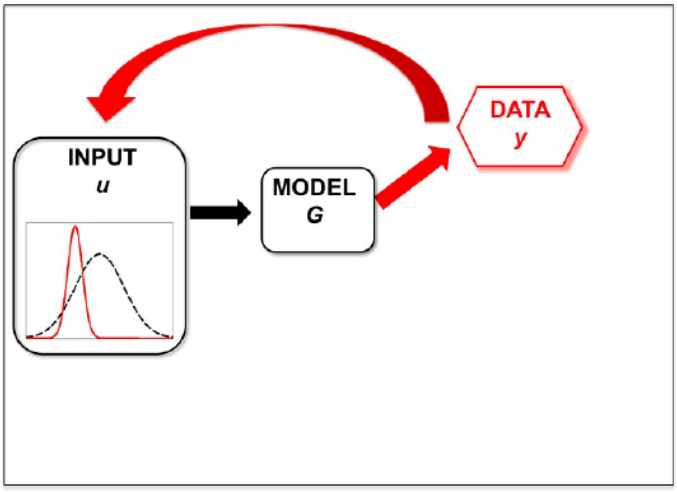


Figure 1.2: Bayesian inverse problem.

map which takes input data $u \in X$ for a partial differential equation into the solution $r \in R$. As Stuart affirms in [Stuart2010], uncertainty quantification in the forward problem framework is concerned with determining the propagation of randomness in the input u into randomness in some *quantity of interest* $q \in Q$, with Q again a Banach space, found by applying the operator $\mathcal{Q} : R \to Q$ to $G(u)$; thus $q = (\mathcal{Q} \circ G)(u)$. The situation is illustrated in Figure 1.1.

Sampling-based techniques are the simplest approaches to propagate uncertainty in numerical simulations: they involve repeated simulation (also called realizations) with a proper selection of the input values. All the results are collected to generate a statistical characterization of the outcome. In the following the Monte Carlo methods, the Multilevel Monte Carlo methods and a comparison between those two will be provided.

1.2 Inverse problem within uncertainty quantification

Given some experimental measurements of a system and some computer simulation results from its mathematical model, inverse uncertainty quantification estimates the discrepancy between the experiment and the mathematical model (which is called bias correction), and estimates the values of unknown parameters in the model if there are any (which is called parameter calibration or simply calibration). Generally this is a much more difficult problem than forward uncertainty propagation; however it is of great importance since it is typically implemented in many model updating process (or "history-matching").

Many inverse problems in the physical sciences require the determination of an unknown

field from a finite set of indirect measurements. Examples include oceanography, oil recovery, water resource management and weather forecasting. In the Bayesian approach to these problems, the unknown and the data are modelled as a jointly varying random variable, typically linked through solution of a partial differential equation, and the solution of the inverse problem is the distribution of the unknown given the data.

Many important results and a lot of work has been done in this field, amongst the most useful is one known as *Bayes' theorem*:

$$\text{prob}(X|Y,I) = \frac{\text{prob}(Y|X,I) \times \text{prob}(X,I)}{\text{prob}(Y,I)} \tag{1.1}$$

The importance of this property to data analysis becomes apparent if we replace X and Y by "input" (also denoted as u) and "data" (also denoted as y):

$$\text{prob}(\textit{input} \mid \textit{data}, I) \propto \text{prob}(\textit{data} \mid \textit{input}, I) \times \text{prob}(\textit{input}, I)$$

The power of Bayes' theorem lies in the fact that it relates the quantity of interest, the probability that the hypothesis is true given the data, to the term we have a better chance of being able to assign, the probability that we would have observed the measured data if the hypothesis was true.

The various terms in Bayes' theorem have formal names. The quantity on the far right, prob($\textit{input}, I$), is called the *prior* probability; it represents our state of knowledge (or ignorance) about the truth of the hypothesis before we have analysed the current data. This is modified by the experimental measurements through the *likelihood* function, or prob($\textit{data} \mid \textit{input}, I$), and yields the *posterior* probability, prob($\textit{input} \mid \textit{data}, I$), representing our state of knowledge about the truth of the hypothesis in the light of the data. In a sense, Bayes' theorem encapsulates the process of learning. We should note, however, that the equality of Eqn 1.1 has been replaced with a proportionality, because the term prob($\textit{data}|I$) (the so-called *evidence*) has been omitted.

This approach is a natural way to provide estimates of the unknown field, together with a quantification of the uncertainty associated with the estimate. It is hence a useful practical modelling tool. However it also provides a very elegant mathematical framework for inverse problems: whilst the classical approach to inverse problems leads to ill-posedness, the Bayesian approach leads to a natural well-posedness and stability theory. Furthermore this framework provides a way of deriving and developing algorithms which are well-suited to the formidable computational challenges which arise from the conjunction of approximations arising from the numerical analysis of partial differential equations.

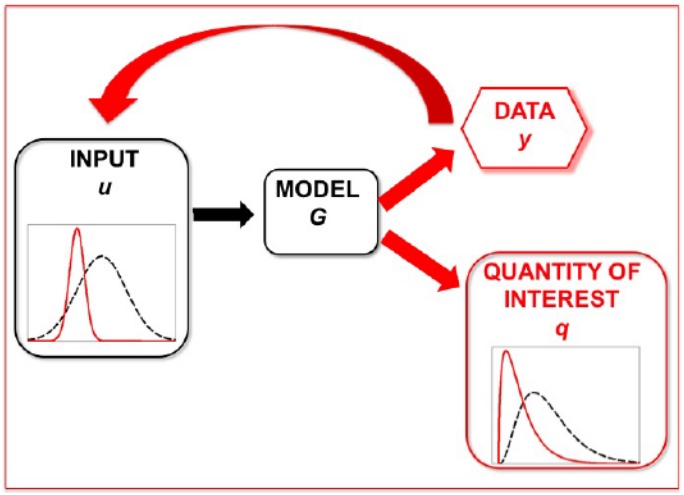


Figure 1.3: Uncertainty quantification in bayesian inversion.

So, in practice, inverse problems are concerned with the problem of determining the input u when given noisy *observed data* y found from $G(u)$. Let Y be the Banach space where the observations lie, let $\mathcal{O} : R \to Y$ denote the *observation operator*, define $\mathcal{G} = \mathcal{O} \circ G$, and consider the equation

$$y = \mathcal{G}(u) + \eta \tag{1.2}$$

viewed as an equation for $u \in X$ given $y \in Y$. The element $\eta \in Y$ represents *noise* and typically something about the size of η is known but the actual instance of η entering the data y is not known. The aim is to reconstruct u from y. The Bayesian inverse problem is to find the conditional probability distribution on $u|y$ from the joint distribution of the random variable (u, y); the latter is determined by specifying the distributions on u and η and, for example, assuming that u and η are independent. This situation is illustrated in Figure 1.2.

To formulate the inverse problem probabilistically it is natural to work with separable Banach spaces as this allows for development of an integration theory as well as avoiding a variety of pathologies that might otherwise arise; we assume separability from now on. The probability measure on u is termed the *prior*, and will be denoted by μ_0, and that on $u|y$ the *posterior*, and will be denoted by μ^y. Once the Bayesian inverse problems has been solved, the uncertainty in q can be quantified with respect to input distributed according to the posterior on $u|y$, resulting in improved quantification of uncertainty in comparison with simply using input distributed according to the prior on u. The situation is illustrated in Figure 1.3. The black dotted lines demonstrate uncertainty quantification prior to incorporating the data, the red

curves demonstrate uncertainty quantification after the data has been incorporated by means of Bayesian inversion.

1.3 Overview

This thesis consists in four chapters and it illustrates methods that could be applicable in both forward and inverse problem frameworks.

In **Chapter 2**, for the sake of generality we introduce the Monte Carlo and Quasi-Monte Carlo method in the context of high-dimensional integration. We show the robustness of the former and the convenience of the latter in terms of convergence rate. Then we show the application of Monte Carlo in the PDEs framework and we provide the relative error analysis. In the end, we introduce another Monte Carlo improving technique, the variance reduction one, which gives the idea that lies behind the main argument of this thesis.

In **Chapter 3**, we show a clever variance reduction technique, the Giles' multilevel Monte Carlo. The theory is provided together with some simple tests cases to show its effectiveness and its convenience in terms of computational costs. In the end we also provide some applications in Computational Fluid Dynamic problems, in particular we provide the results of the simulation of PDEs on a random geometry. So, if in a first instance we deal with PDEs with random coefficient coming from a Karhunen-Loève expansion or from a piecewise constant model in the second one we try to apply the same methodologies to a different kind of random parameter.

CHAPTER 2

Monte Carlo Method

2.1 Introduction

In this chapter we first review the Monte Carlo and the quasi-Monte Carlo method in the context of high-dimensional integration. Then we show the application of Monte Carlo in PDE with random coefficients and, after some error analysis, we provide and comment some numerical tests with their results.

The Monte Carlo method is a widely used tool in many disciplines, including physics, chemistry, engineering, finance, biology, computer graphics, operations research and management science. Examples of problems that it can address are:

- A call center manager wants to know if adding a certain number of service representative during peak hours would help decrease the waiting time of calling customers.
- A portfolio manager needs to determine the magnitude of the loss in value that could occur with a 1% probability over a one-week period.
- The designer of telecommunications network needs to make sure that the probability of losing information cells in the network is below a certain thresold.

Realistic models of the system above typically assume that at least some of their components behave in a random way. For instance, the call arrival times and processing times for the call center cannot realistically be assumed to be fixed and known ahead of time and thus it makes sense instead to assume that they occur according to some stochastic model.

The Monte Carlo simulation method uses random sampling to study properties of systems with components that behave in a random fashion. More precisely, the idea is to *simulate* on the computer the behavior of these systems by randomly generating the variables describing the behavior of their components. Samples of the quantities of interest can then be obtained and used for statistical inference.

2.2 Monte Carlo method

The Monte Carlo method is certainly very popular and its origins can be traced back in 1946, when physicists at Los Alamos Scientific Laboratory were investigating radiation shielding and the distance that neutrons would likely travel through various materials. Despite having most of the necessary data, such as the average distance a neutron would travel in a substance before it collided with an atomic nucleus, and how much energy the neutron was likely to give off following a collision, the Los Alamos physicists were unable to solve the problem using conventional, deterministic mathematical methods. Stanislaw Ulam had the idea of using random experiments. He recounts his inspiration when he was convalescing from an illness and playing solitaires. He questioned himself on the chances that the cards of a Canfield solitaire can come out successfully. After spending a lot of time trying to estimate them by pure combinatorial calculations, he wondered whether a more practical method than “abstract thinking” might not be to lay it out say one hundred times and simply observe and count the number of successful plays. This was already possible to envisage with the beginning of the new era of fast computers, and he immediately thought of problems of neutron diffusion and other questions of mathematical physics, and more generally how to change processes described by certain differential equations into an equivalent form interpretable as a succession of random operations. Later in 1946, he described the idea to John von Neumann and they began to plan actual calculations.

Being secret, the work of von Neumann and Ulam required a code name. Von Neumann chose the name Monte Carlo. The name refers to the Monte Carlo Casino in Monaco where Ulam’s uncle would borrow money to gamble. Using lists of "truly random" random numbers was extremely slow, but von Neumann developed a way to calculate pseudorandom numbers, using the middle-square method. Though this method has been criticized as crude, von Neumann was aware of this: he justified it as being faster than any other method at his disposal, and also noted that when it went awry it did so obviously, unlike methods that could be subtly incorrect.

Monte Carlo methods were central to the simulations required for the Manhattan Project, though severely limited by the computational tools at the time. In the 1950s they were used at Los Alamos for early work relating to the development of the hydrogen bomb, and became popularized in the fields of physics, physical chemistry, and operations research. The Rand Corporation and the U.S. Air Force were two of the major organizations responsible for funding and disseminating information on Monte Carlo methods during this time, and they began to find a wide application in many different fields.

So, the fundamental idea on which Monte Carlo (MC) methods rely is a pseudo-random sampling of a RV in order to construct a set of realization of the input data. To each of these realizations corresponds a unique solution of the model.

In many stochastic applications one wants to estimate $\mathbb{E}[Y]$. In standard MC approach one can simulate it using

$$\mathscr{A}(Y;M) \equiv \frac{1}{M}\sum_{i=1}^{M} Y(\omega_i),$$

ω_i being independent and identically distributed (iid) samples, and choose M sufficiently large to control the *statistical error*

$$\mathbb{E}[Y] - \mathscr{A}(Y;M)$$

The following example of Monte Carlo integration will be more clarifying than any other explanations.

Example *The integral $I = \int_{[0,1]^N} f(\mathbf{x})d\mathbf{x}$ will be computed by the Monte Carlo method, where it is assumed $f(\mathbf{x}) : [0,1]^N \to \mathbb{R}$. Let $Y = f(X)$, where X is uniformly distributed in $[0,1]^N$. One has:*

$$\begin{aligned} I &= \int_{[0,1]^N} f(x)dx \\ &= \int_{[0,1]^N} f(x)p(x)dx \qquad [p \text{ is the uniform pdf}] \\ &= \mathbb{E}\big[f(x)\big] \qquad\qquad [x \text{ is uniformly distributed in } [0,1]^N] \\ &\simeq \frac{1}{M}\sum_{i=1}^{M} f(x(\omega^{(i)})) \\ &\equiv I_{N,M} \end{aligned}$$

where the values $\{\mathbf{x}(\omega_j)\}$ are iid and are sampled uniformly in the cube $[0,1]^d$ by sampling the components $x_i(\omega_j)$ independently and uniformly on the interval $[0,1]$. We can find an example on how the evaluation points are taken in Figure 2.1, where we can see the difference between a Monte Carlo quadrature against a deterministic one (uniform).

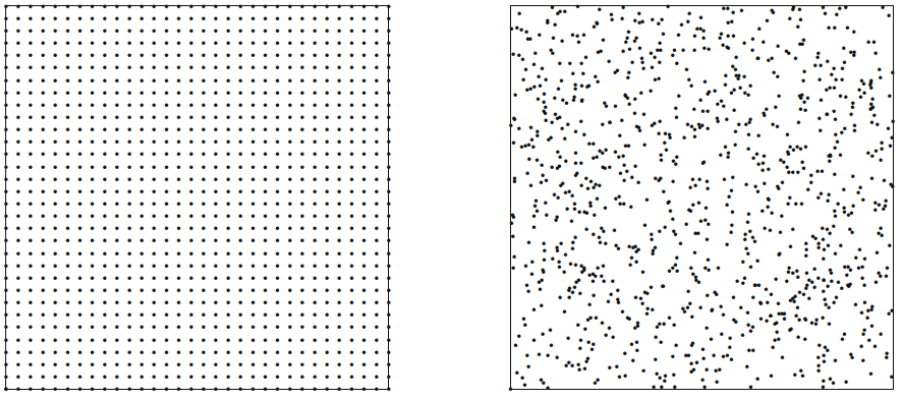


Figure 2.1: Deterministic quadrature (left) vs. Monte Carlo sampled (right) points in the case $d = 2$.

What ensure the convergence of the method are the laws of large numbers that we recall briefly in the following. First of all, we introduce the weak law of large numbers.

Theorem 1 (Weak law of large numbers)**.** *Assume Y_j, $j = 1,2,3,\ldots$ are independent, identically distributed random variables and $\mathbb{E}\left[Y_j\right] = \mu < \infty$. Then*

$$\sum_{j=1}^{M} \frac{Y_j}{M} \xrightarrow{P} \mu, \quad \text{as } M \to \infty, \tag{2.1}$$

where $\xrightarrow{P}$ denotes convergence in probability, i.e. the convergence 2.1 means

$$P\big(|\sum_{j=1}^{M} \frac{Y_j}{M} - \mathbb{E}[Y]| > \epsilon\big) \to 0$$

for all $\epsilon \to 0$

Then, by changing the definition of convergence we can state the strong law of large numbers.

Theorem 2 (Strong law of large numbers)**.** *Assume Y_j, $j = 1,2,3,\ldots$ are independent, identically distributed random variables with $\mathbb{E}\left[|Y_j|\right] < \infty$ and $\mathbb{E}\left[Y_j\right] = \mu$. Then*

$$\sum_{j=1}^{M} \frac{Y_j}{M} \xrightarrow{a.s.} \mu, \quad \text{as } M \to \infty, \tag{2.2}$$

where $\xrightarrow{a.s.}$ denotes almost sure convergence, i.e. the convergence 2.2 means

$$P\big(\{\sum_{j=1}^{M} \frac{Y_j}{M} \to \mu\}\big) = 1.$$

In other words, the important result that lies behind the Laws of Large Numbers is that a for large M the empirical average is very close to the expected value μ with very high probability.

Now that we know for sure that the method converge to something, we would like to say something about the rate of convergence. To understand this and how the statistical error behave we can refer to the Central Limit Theorem that now will be also recalled.

Theorem 3. *Assume ξ_j, $j = 1,2,3,\ldots$ are independent, identically distributed (iid) and $\mathbb{E}\left[\xi_j\right] = 0$, $\mathbb{E}\left[\xi_j^2\right] = 1$. Then*

$$\sum_{j=1}^{M} \frac{\xi_j}{\sqrt{M}} \rightharpoonup \nu,$$

where ν is $N(0,1)$ and $\rightharpoonup$ denotes convergence of the distributions, also called weak convergence, i.e. the convergence means

$$\mathbb{E}\left[g\Big(\sum_{j=1}^{M} \frac{\xi_j}{\sqrt{M}}\Big)\right] \to \mathbb{E}\left[g(\nu)\right],$$

for all bounded and continuous functions g.

Having this in mind and referring to the example, one can compute the error $I_M - I$ to see in practice how this theorem can be applied in the MC context.

Let the error ϵ_M be defined by

$$\begin{aligned} \epsilon_M &= \sum_{j=1}^{M} \frac{f(x_j)}{M} - \int_{[0,1]^d} f(x)dx \\ &= \sum_{j=1}^{M} \frac{f(x_j) - M\mathbb{E}\left[f(x)\right]}{M} \end{aligned}$$

By the Central Limit Theorem, one has

$$\sqrt{M}\epsilon_M \rightharpoonup \sigma\nu,$$

where ν is $N(0,1)$ and

$$\begin{aligned} \sigma^2 &= \int_{[0,1]^d} f^2(x)dx - \Big(\int_{[0,1]^d} f(x)dx\Big)^2 \\ &= \int_{[0,1]^d} \Big(f^2(x) - \int_{[0,1]^d} f(x)dx\Big)^2 dx \end{aligned}$$

In practice, σ^2 is approximated by

$$\hat{\sigma}^2 = \frac{1}{M-1}\sum_{j=1}^{M}\Big(f(x_j) - \sum_{m=1}^{M}\frac{f(x_m)}{M}\Big)$$

This implies that for any set $B = (-C, C)$

$$P(\sqrt{M}\epsilon_M \in B) \to P(N(0,1) \in B)$$

Given a constant, $0 < \alpha \ll 1$, one has to choose $C = C_\alpha$ such that the following confidence level constraint is satisfied

$$P(N(0,1) \in B) = \int_{|x| \le C_\alpha} \frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}} dx = 1 - \alpha$$

So in the end one has

$$P(|\epsilon_M| \le \frac{C_\alpha}{\sqrt{M}}) \approx 1 - \alpha, \qquad \text{for large } M$$

Hence the statistical error of the Monte Carlo estimator is $O(1/\sqrt{M})$, which is independent of the dimension d. The comparison between the convergence rate of a deterministic quadrature, say $M^{-2/d}$ of the trapezoidal rule, versus $M^{-1/2}$ supports the suggestion that, even for moderate dimensions d, the Monte Carlo method can outperform deterministic methods.

Although the Monte Carlo error has the nice property that its convergence rate of $1/\sqrt{M}$ does not depend on the dimension, this rate is slow. For this reason, a lot of work has been done to find ways of improving the Monte Carlo error and two different paths can be taken. The first one is try to find ways of reducing the variance σ^2 of f. Methods achieving this are under the category of *variance reduction techniques*. The second approach is to use an alternative sampling mechanism (a deterministic one actually) - often called *quasi-random* or *low-discrepancy* sampling - whose corresponding error has a better convergence rate. Using these alternative sampling mechanisms for numerical integration is usually referred to as *quasi-Monte Carlo* integration.

2.2.1 Numerical tests

In this paragraph we show the results of some actual computations. The sources of these exercises can be traced back at the summer school in Mathematical and Algorithmic aspects of Uncertainty Quantification hold by Nobile and Tempone at the University of Texas, Austin. Similarly to the very first example of the chapter, we are interested in calculating the following quantity

$$g = \int_{[0,1]^N} f(\mathbf{x}) d\mathbf{x}$$

for some different functions that mainly differ by the degree of regularity. In particular we look at different examples of f for some real constants $\{c_n, w_n\}_{n=1}^N$ taken from

1. Gaussian: $f(\mathbf{x}) = \exp(\sum_{n=1}^{N} c_n^2 (x_n - w_n)^2)$, with $c_n = 7.03/N$ and $w_n = \frac{1}{2}$.
   The exact solution reads:
$$\int_{[0,1]^N} f(\mathbf{x})\, d\mathbf{x} = \prod_{n=1}^{N} \frac{\sqrt{\pi}}{2c_n} (\mathrm{erf}(c_n(1-w_n)) + \mathrm{erf}(c_n w_n))$$

2. Continuous: $f(\mathbf{x}) = \exp(-\sum_{n=1}^{N} c_n |x_n - w_n|)$, with $c_n = 2.04/N$ and $w_n = \frac{1}{2}$.
   The exact solution reads:
$$\int_{[0,1]^N} f(\mathbf{x})\, d\mathbf{x} = \prod_{n=1}^{N} \frac{1}{c_n} (2 - e^{-c_n w_n} - e^{-c_n(1-w_n)})$$

3. Discontinuous: $f(\mathbf{x}) = \begin{cases} 0, & \text{if } x_1 > w_1 \text{ or } x_2 > w_2 \\ \exp(-\sum_{n=1}^{N} c_n x_n), & \text{otherwise} \end{cases}$ with $c_n = 4.3/N$ and $w_1 = \frac{\pi}{4}$ and $w_2 = \frac{\pi}{5}$.
   The exact solution reads:
$$\int_{[0,1]^N} f(\mathbf{x})\, d\mathbf{x} = \frac{1}{\prod_{n=1}^{N} c_n} (e^{c_1 w_1} - 1)(e^{c_2 w_2} - 1) \prod_{n=3}^{N} (e^{c_n} - 1).$$

The pseudo code used for each case is straightforward

Algorithm 1 MonteCarlo

1: $u \leftarrow$ random vector of dimension $M \times N$ [M : #samples, N : dimension of the problem]
2: compute $f(u)$
3: $I \leftarrow \frac{1}{M} \sum_{m=1}^{M} f(u_m)$
4: compare I with I_{exact}

From this simple test we want to show the effectiveness and the robustness of the method. As we can see in Figure 2.2, the regularity of the function is not affecting the results but the convergence is very slow and goes with $1/\sqrt{M}$. The Central Limit Theorem bound, the red line in the figure, is computed as $c_0 \hat{\sigma}/\sqrt{M}$, where $c_0 = 3$ and $\hat{\sigma}$ is the sample variance

$$\hat{\sigma}^2 = \frac{1}{M-1} \sum_{j=1}^{M} \Big(f(x_j) - \sum_{m=1}^{M} \frac{f(x_m)}{M} \Big)^2$$

The computational error, depicted in black, is computed as a function of M and it is calculated as the cumulative sum of the absolute value of the difference between the exact solution and the computational one divided by the samples M. This error is below the red line with a certain probability depending on the choice of c_0 (related to the normal distribution).

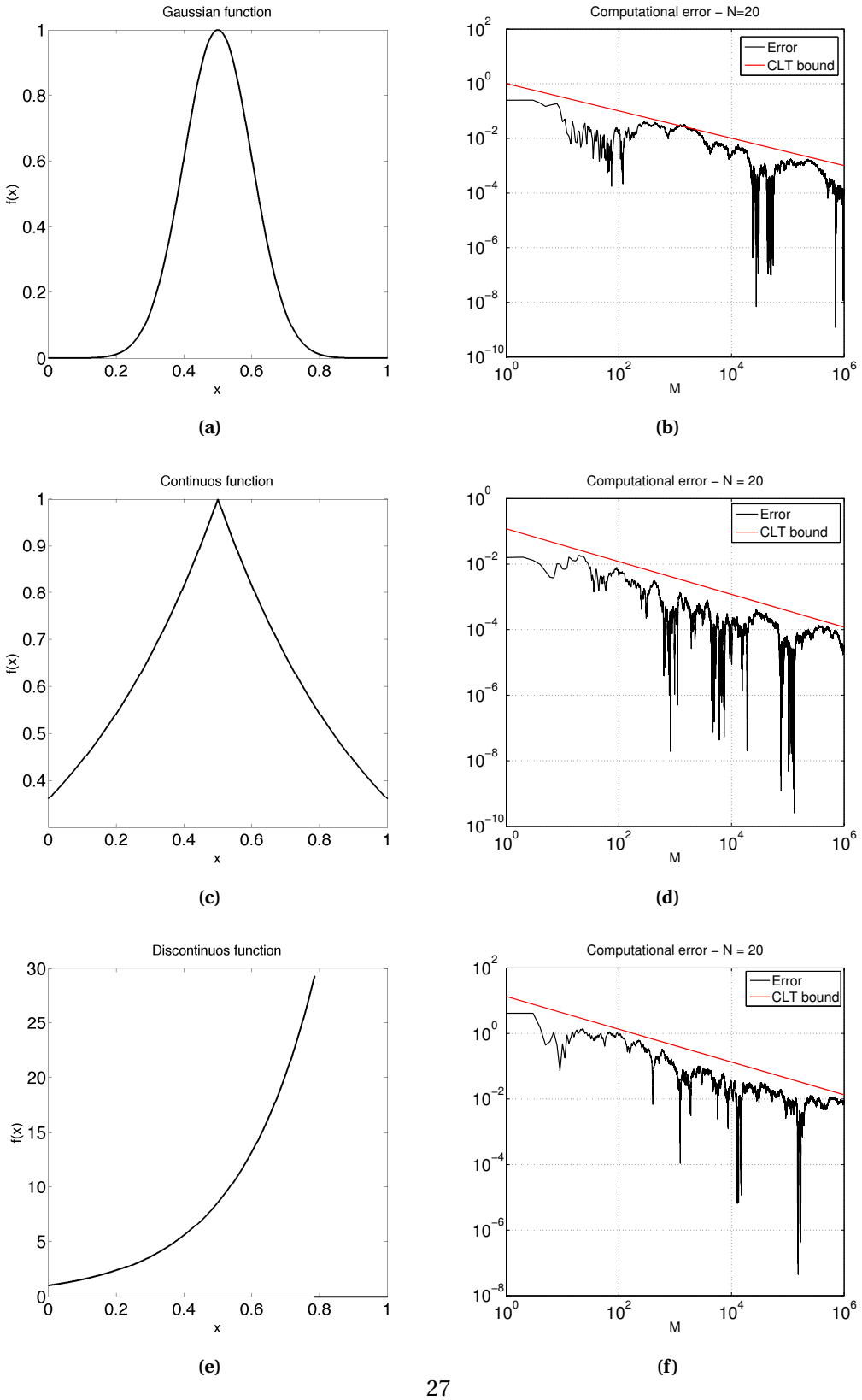


Figure 2.2: 1D functions used to test the MC integration (left column) and convergence of the computational error as $1/\sqrt{M}$ (right column).

2.3 Quasi-Monte Carlo method

This section discusses alternatives to Monte Carlo simulation known as *quasi-Monte Carlo* or *low-discrepancy* methods. These methods differ from ordinary Monte Carlo in that they make no attempt to mimic randomness. Indeed, they seek to increase accuracy specifically by generating points that are too evenly distributed to be random. This contrasts with the variance reduction techniques that will be discussed in the next sections, which take advantage of the stochastic formulation to improve precision.

Low-discrepancy methods have the potential to accelerate convergence from the $O(1/\sqrt{M})$ rate associated with Monte Carlo (M being the number of paths or points generated) to nearly $O(1/M)$ convergence: under appropriate conditions, the error in a quasi-Monte Carlo approximation is $O(1/M^{1-\varepsilon})$ for all $\varepsilon > 0$. Standard variance reduction techniques, affecting only the implicit constant in $O(1/\sqrt{M})$, are not nearly so ambitious. We will see, however, that the ε in $O(1/M^{1-\varepsilon})$ hides a dependence on problem dimension.

The tools used to develop and analyze low-discrepancy methods are very different from those used in ordinary Monte Carlo, as they draw on number theory and abstract algebra rather than probability and statistics. Our goal is therefore to present key ideas and methods rather than an account of the underlying theory. Moreover, the QMC approach has been well surveyed in a recent work by Dick, Kuo and Sloan[1].

So we are now going to introduce the use of *low-discrepancy* sampling to replace the pure random sampling that forms the backbone of the Monte Carlo method. A low-discrepancy sample is one whose points are distributed in a way that approximates the uniform distribution as closely as possible. Unlike for random sampling, points are not required to be independent. In fact, the sample might be completely deterministic.

As a first step, let us consider $P = \{\xi_1, \ldots, \xi_M\}$ a set of points $\xi_i \in [0,1]^N$ and $f : [0,1]^N \to \mathbb{R}$ a continuos function. A quasi-Monte Carlo method (QMC) to approximate $I_N(f) = \int_{[0,1]^N} f(\mathbf{y}) d\mathbf{y}$ is an equal weight cubature formula of the form

$$I_{N,M}(f) = \frac{1}{M} \sum_{i=1}^{M} f(\xi_i)$$

Let us now introduce a way of measuring the uniformity of a point set that is not specific to a particular type of construction. More precisely, the idea is to measure the distance between the empirical distribution induced by the point set and the uniform distribution via

[1] J. Dick, F. Y. Kuo, I. H. Sloan *High-dimensional integration: The quasi-Monte Carlo way*, Acta Numerica / Volume 22 / May 2013, pp 133-288.

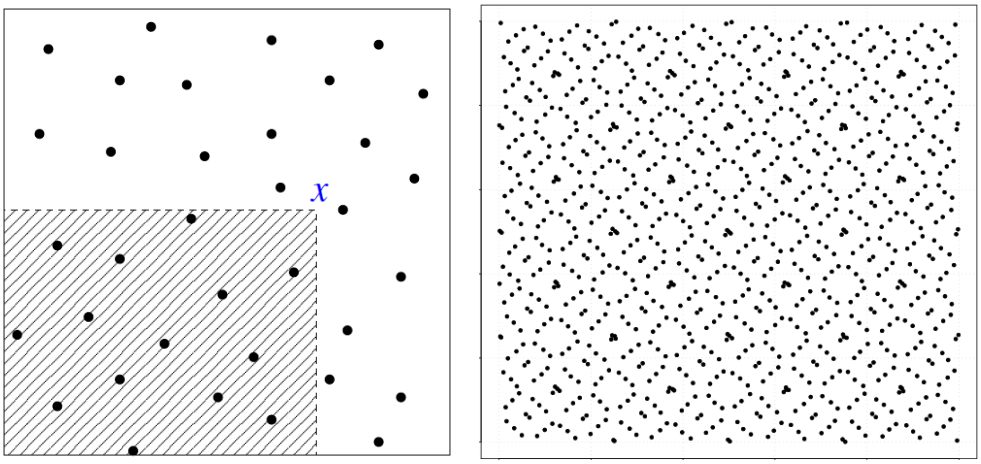
the Kolmogorov-Smirnov statistic. The concept of *discrepancy* looks precisely at such distance measures. We can then define it as

Definizione 1. *Let* $\mathbf{x} \in [0,1]^N$ *and* $[\mathbf{0},\mathbf{x}] = [0,x_1] \times \cdots \times [0,x_N]$, *then the* local discrepancy *is*

$$\Delta_P(\mathbf{x}) = \frac{1}{M}\sum_{i=1}^{M} \mathbf{1}_{[\mathbf{0},\mathbf{x}]}(\xi_i) - \prod_{i=1}^{N} x_i$$

while the star-discrepancy *is*

$$\Delta^*_{P,N} = \sup_{\mathbf{x}\in[0,1]^N} |\Delta_P(\mathbf{x})|$$



(a) *Discrepancy.* **(b)** *Sobol sequence.*

Figure 2.3: Idea of discrepancy (left) and example of Sobol sequence (right) in a two-dimensional domain. The discrepancy can be also seen more intuitively as $\Delta_P = \frac{\#\text{ points in } [\mathbf{0},\mathbf{x}]}{M} - \text{Vol}([\mathbf{0},\mathbf{x}])$.

We can have a more intuitive idea of discrepancy by looking at Figure 2.3(a) and we can give an alternative definition as

$$\Delta_P = \frac{\#\text{ points in } [\mathbf{0},\mathbf{x}]}{M} - \text{Vol}([\mathbf{0},\mathbf{x}])$$

At this point we can introduce the definition of low discrepancy sequences.

Definizione 2. *A sequence* $(\xi_1,\xi_2,\ldots)$, $\xi_i \in [0,1]^N$ *is a* low discrepancy sequence *if the set of points* $P_M = (\xi_1,\ldots,\xi_M)$ *satisfies*

$$\Delta^*_{P_M,N} \leq C_N \frac{(\log M)^N}{M}.$$

So we can say that QMC methods use equal weight cubature formulae with low discrepancy sequences of points. Several sequences exist in literature, some of them available also in MATLAB, such as: Halton, Hammersley, Sobol, Faure, Niederreiter, etc...

The simplest example is the Halton sequence and in order to construct that sequence we first have to define the *radical inverse function* $\phi_b(i)$ as follows:

$$\text{if } i = \sum_{n=1}^{\infty} i_n b^{n-1}, \quad i_n \in \{0,1,\dots,b-1\}, \qquad \text{then } \phi_b(i) = \sum_{n=1}^{\infty} i_n b^{-n},$$

so, in base $b = 10$, the radical inverse of $i = 15421$ is $\phi_{10}(i) = 0.12451$. Then, if $p_1,\dots,p_N$ denote the first N prime numbers, the Halton sequence $P = \{\xi_1,\xi_2,\dots\}$ is given by

$$\xi_i = (\phi_{p_1}(i),\phi_{p_2}(i),\dots,\phi_{p_N}(i))$$

and its star-discrepancy satisfies

$$\Delta_N^* := \sup_{\mathbf{x}\in[0,1]^N} |\Delta_P(\mathbf{x})| = O\Big(\frac{(\log M)^N}{M}\Big)$$

Other better sequences instead such as Hammersley, Sobol, Faure, etc... have a star-discrepancy

$$\Delta_N^* = O\Big(\frac{(\log M)^{N-1}}{M}\Big).$$

2.3.1 Error analysis in 1D

Let us consider for simplicity the $N = 1$ dimensional case just to have an idea on how the discrepancy comes into play in this framework. The result can be generalized for arbitrary dimension. The following *error representation* holds:

$$\frac{1}{M}\sum_{i=1}^{M} f(\xi_i) - \int_{[0,1]} f(y)\,dy = \Delta_P(1)f(1) - \int_{[0,1]} \Delta_P(y)f'(y)\,dy$$

which proof can be find in [Nobile,Tempone].

Hence, the error in the QMC integration is bounded by

$$\Big|\frac{1}{M}\sum_{i=1}^{M} f(\xi_i) - \int_{[0,1]} f(y)\,dy\Big| = \Big|\int_{[0,1]} \Delta_P(y)f'(y)\,dy\Big| \le \Delta_P^* \|f'\|_{L^1(0,1)}$$

To estimate the error in practice in a quasi-Monte Carlo computation, the following strategy has been adopted. We consider $\eta \sim U([0,1]^N)$ as a uniformly distributed random vector (shift) and

$$I_{N,M}(f,\eta) = \frac{1}{M}\sum_{i=1}^{M} f(\{\xi_i + \eta\})$$

as the QMC formula with the random shift η. Then we take s i.i.d. shifts $\eta_1, \ldots, \eta_s \sim U([0,1]^N)$. Finally we estimate the integral

$$I_{N,M}(f) = \frac{1}{s}\sum_{j=1}^{s} I_{N,M}(f,\eta_j) = \frac{1}{sM}\sum_{j=1}^{s}\sum_{i=1}^{M} f(\{\xi_i + \eta_j\})$$

and the error based on the $\eta-$sample standard deviation, i.e.

$$e_{N,M}(f) = \frac{c_0}{\sqrt{s}}\sqrt{\frac{1}{\sqrt{s-1}}\sum_{j=1}^{s}(I_{N,M}(f,\eta_j) - I_{N,M}(f))^2}$$

In Figure 2.4 this quantity is represented with a green label. We can see that it gets more and more precise as the number of shifts increase[2].

[2] More details about these results can be found in [Nobile,Tempone].

2.3.2 Numerical tests

In this section we provide an estimation of the same integrals reported above with a quasi-Monte Carlo method. We use the function `i4_sobol_generate.m` to generate Sobol sequences[3].

The pseudocode for the algorithm is the following

Algorithm 2 Quasi Monte Carlo

1: $U \leftarrow$ i4_sobol_generate(M, N) [generate the Sobol sequence]
2: **for** $i = 1 \rightarrow S$ **do**
3: shift $\leftarrow$ rand$(N, 1)$, [generate the random vector of shifts]
4: $u_{\text{shift}} \leftarrow \text{mod}(U + SS, 1)$ [compute a shifted sequence]
5: $u \leftarrow [u, u_{\text{shift}}]$ [update the overall vector of random points]
6: compute $f(u)$
7: $I \leftarrow \frac{1}{M \cdot S} \sum_{m=1}^{M \cdot S} f(u_m)$
8: compare I with I_{exact}

The Sobol sequence generator works on base 2 so the initial amount of sample is $M = 2048$. Moreover, for the computation a number of shifts $s = 1000$ is chosen.

In Figure 2.4 we can see in the right column the behavior of the correspondent function on the left. In the black line we indicate the standard Monte Carlo computational error as we did in the previous section. in the blue line instead, we represent the Quasi-Monte Carlo computational error. The red lines represent the two rates $O(1/\sqrt{M})$ and $O(1/M)$ while the green one is the error based on the $\eta-$sample standard deviation. We were expecting the latter to be above the QMC error with a certain probability related to the choice of c_0, but as we can see, is true only after a certain number of shifts.

It is clear how in this framework the methods does not behave in the same way for different types of regularity of the integrand function. If in the case of a continuos or a infinitely differentiable function we have a gain of the rate at almost $O(1/M)$ in the case of the discontinuous function instead remain pretty similar to the stdMC one.

[3] This package has been taken from John Burkardt's home page:
`http://people.sc.fsu.edu/~jburkardt/m_src/m_src.html`.

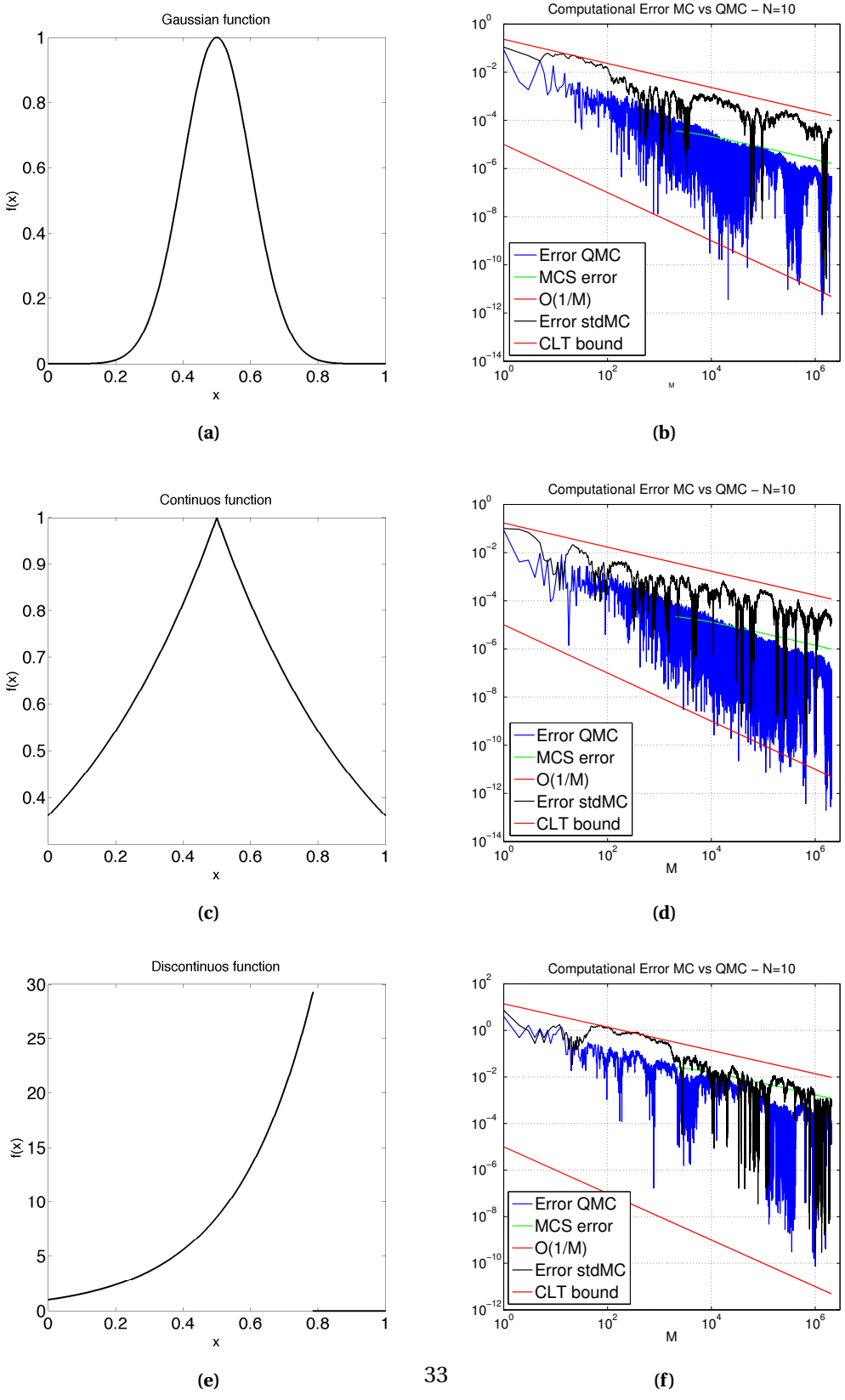


Figure 2.4: 1D functions used to test the MC integration (left column) and convergence of the computational error (right column) as $1/\sqrt{M}$ for the MC and $1/M$ for the "good cases" of QMC.

2.4 Monte Carlo method for PDEs with random coefficient

In this section we provide some numerical analysis to discuss the sources of error that occurs when we deal with SPDEs and we try to solve the randomness with a sampling-based technique. We are going to define and denote u as the solution of a certain problem, $\mathbf{y}$ as the vector of random parameters and we want to find the expectation of a certain quantity of interest Q in order to place ourselves in the framework described in Chapter 1. The main results of this analysis come from [32]. Here we develop some calculations and add some clarifications.

Let $\mathbf{y} = (y_1, \dots, y_N)$ be a random vector with density $\rho(\mathbf{y}) : \Gamma \to \mathbb{R}_+$, $u(\mathbf{y}) : \Gamma \to V$ a Hilbert-valued function, $u \in L^2_\rho(\Gamma; V)$, and $Q : V \to \mathbb{R}$ a continuous functional on V (possibly non linear), such that $\mathbb{E}\left[|Q(u(\mathbf{y}))|^p\right] < \infty$ for p sufficiently large.

Similarly as before, the goal here is to compute $\mathbb{E}\left[Q(u(\mathbf{y}))\right]$ and using the classical Monte Carlo approach one can approximate expectation by sample averages. In fact, if $\{\mathbf{y}(\omega_m)\}_{m=1}^M$ be iid $\mathbf{y}$ samples one has

$$\mathbb{E}\left[Q(u(\mathbf{y}))\right] \approx \frac{1}{M}\sum_{m=1}^{M} Q(u(\mathbf{y}(\omega_m)))$$

where for each $\mathbf{y}(\omega_m)$ one has to find the solution $u(\mathbf{y}(\omega_m))$ of the PDE and evaluate the q.o.i. $Q(u(\mathbf{y}(\omega_m)))$. As said before, the Monte Carlo approach has the nice property that the convergence rate is $M^{-1/2}$ independently from the length of $\mathbf{y}$ and the regularity of $u(.)$, but the convergence is slow and, in this case, the possibly available regularity of the solution cannot be exploited.

Assume now that the PDE had been discretized by some mean (finite elements, finite volumes, spectral methods,...), so that in practice the discrete solutions $u_h(\mathbf{y}(\omega_m))$, $m = 1, \dots, M$ are computed. Then the MC estimator will be

$$\mathbb{E}\left[Q(u(\mathbf{y}))\right] \approx \frac{1}{M}\sum_{m=1}^{M} Q(u_h(\mathbf{y}(\omega_m)))$$

2.4.1 Error analysis

At this point some error analysis can be done. The first thing that one can notice is that the error is composed of two terms. In fact

$$\mathbb{E}\left[Q(u(\mathbf{y}))\right] - \frac{1}{M}\sum_{m=1}^{M} Q(u_h(\mathbf{y}(\omega_m))) =$$

$$= \mathbb{E}\left[Q(u(\mathbf{y}))\right] - \mathbb{E}\left[Q(u_h(\mathbf{y}))\right] + \mathbb{E}\left[Q(u_h(\mathbf{y}))\right] - \frac{1}{M}\sum_{m=1}^{M} Q(u_h(\mathbf{y}(\omega_m))) =$$

$$= \underbrace{\mathbb{E}\left[Q(u(\mathbf{y})) - Q(u_h(\mathbf{y}))\right]}_{\text{discretization error}} + \underbrace{\mathbb{E}\left[Q(u_h(\mathbf{y}))\right] - \frac{1}{M}\sum_{m=1}^{M} Q(u_h(\mathbf{y}(\omega_m)))}_{\text{statistical error}}$$

where the first term is the *discretization error* that will be denoted as $\mathscr{E}^Q(h)$ while the second is the *statistical error* that will be denoted $\mathscr{E}_h^Q(M)$.

Discretization error

Let $Q : V \to \mathbb{R}$ with $Q(0) = 0$ be a functional that is globally Lipschitz, i.e.

$$\exists C_Q > 0 \quad \text{s.t.} \quad |Q(u) - Q(u')| \leq C_Q \|u - u'\|_V, \quad \forall u, u' \in V$$

and assume that exists $\alpha > 0$ and $C_u(\mathbf{y}) > 0$ with $\int_\Gamma C_u(\mathbf{y})^p \rho(\mathbf{y}) d\mathbf{y} < \infty$ for some $p > 1$, such that

$$\|u(\mathbf{y}) - u_h(\mathbf{y})\|_V \leq C_u(\mathbf{y}) h^\alpha, \qquad \forall \mathbf{y} \in \Gamma \text{ and } 0 < h < h_0 \tag{2.3}$$

Then

$$\begin{aligned} |\mathscr{E}^Q(h)| = |\mathbb{E}\left[Q(u(\mathbf{y})) - Q(u_h(\mathbf{y}))\right]| &= |\int_\Gamma (Q(u(\mathbf{y})) - Q(u_h(\mathbf{y})))\rho(\mathbf{y}) d\mathbf{y}| \\ &\leq C_Q \int_\Gamma \|u(\mathbf{y}) - u_h(\mathbf{y})\|_V \rho(\mathbf{y}) d\mathbf{y} \\ &\leq C_Q h^\alpha \int_\Gamma C_u(\mathbf{y}) \rho(\mathbf{y}) d\mathbf{y} \\ &\leq C_Q \|C_u\|_{L^p_\rho} h^\alpha \end{aligned}$$

The value of α in Equation 2.3 may depend both on the space V and on the accuracy order of the approximation method chosen. Since in our application we take both piecewise linear finite element method and first order accuracy finite volume methods and the space is usually L^2 the we are expecting $\alpha = 2$.

Example *Consider the following elliptic problem with uniformly bounded random coefficient*

$$\begin{cases} -\text{div}(a(x,\mathbf{y})\nabla u(x,\mathbf{y})) = f(x), & x \in D \\ u(x,\mathbf{y}) = 0, & x \in \partial D \end{cases} \qquad \forall \mathbf{y} \in \Gamma \subset \mathbb{R}^N$$

where $D \subset \mathbb{R}^d$ is an open, convex, Lipschitz domain and $f \in L^2(D)$. The assumptions on the random coefficient are the following:

- *finite Karhunen–Loève expansion: $a(x,\mathbf{y}) = \bar{a} + \sum_{i=1}^N \sqrt{\lambda_i} y_i b_i(x)$;*

- *independence of the RVs:* $y_i \sim \mathcal{U}(-\sqrt{3},\sqrt{3})$ *iid (so* $\Gamma = [-\sqrt{3},\sqrt{3}]^N$*);*
- $b_i \in C^\infty(D)$;
- $\sum_{i=1}^N \sqrt{3\lambda_i}\|b_i\|_\infty \le \delta\bar{a}$, $0<\delta<1$.

Thanks to these one can state the following

$$(1-\delta)\bar{a} \le a(x,\mathbf{y}) \le (1+\delta)\bar{a}, \qquad \|\nabla a(\cdot,\mathbf{y})\|_{L^\infty(D)} \le C_a, \qquad \forall \mathbf{y}\in\Gamma$$

that ensure the coercivity of the bilinear form. Moreover, denoting

$$H_0^1 = \{v \in H^1(D) : \|v-\varphi_n\|_{H^1(D)} \to 0, \text{ for some } (\varphi_n)\subset C_0^\infty(D)\}$$

endowed with the norm $||v||_{H_0^1} = ||\nabla v||_{L^2(D)}$ *and thanks to the Lax-Milgram theorem (see [7], sect. 4.2) which is valid under these conditions, we can assert the existence, the uniqueness of the solution* $u(\mathbf{y}) \in H_0^1 \subset H^1(D)$ *and the continuos dependency from the data*

$$\|u(\mathbf{y})\|_{H_0^1(D)} \le \frac{C_P\|f\|_{L^2(D)}}{(1-\delta)\bar{a}}, \qquad \forall \mathbf{y}\in\Gamma$$

where C_P *is the Poincaré constant, i.e.*

$$\|v\|_{L^2(D)} \le C_P\|v\|_{H_0^1(D)}, \qquad v \in H_0^1(D).$$

Moreover, since the problem is elliptic with homogenous boundary conditions, $f \in L^2$ *and the domain is convex (see [8]), a* uniform bound *in* $\mathbf{y}$ *can be stated on the* H^2-*norm of the solution*

$$\exists C_u > 0, \quad s.t. \quad \|u(\mathbf{y})\|_{H^2(D)} \le C_u, \qquad \forall \mathbf{y}\in\Gamma$$

Since the problem is well-posed in the sense of Hadamard, one can state the weak formulation and then consider a well-posed discrete formulation. Now in particular, a piecewise linear finite element approximation will be taken in consideration so the problem can be written

$$\begin{cases} \forall \mathbf{y}\in\Gamma \text{ find } u_h(\mathbf{y}) \in V_h, \text{ s.t.} \\ \int_D a(\cdot,\mathbf{y})\nabla u_h(\mathbf{y})\cdot\nabla v_h(\mathbf{y}) = \int_D f v_h \qquad \forall v_h \in V_h \end{cases}$$

where $V_h \subset H_0^1(D)$ *is the space of continuous piecewise linear functions on a* uniform *and* admissible *triangulation* $\mathcal{T}_h$*, vanishing on* ∂D*.*

The discrete solution $u_h(\mathbf{y})$ *satisfies the same bound as the continuous one,*

$$||u_h(\mathbf{y})||_{H^1(D)} \le \frac{||f||_{L^2(D)}}{(1-\delta)\bar{a}\sqrt{1+C_P^2}}, \qquad \forall \mathbf{y}\in\Gamma$$

and

$$\mathbb{E}\left[|Q(u_h(\mathbf{y}))|^p\right]^{\frac{1}{p}} \leq \mathbb{E}\left[(C_Q||u_h(\mathbf{y})||)^p\right]^{\frac{1}{p}} \leq \frac{C_Q||f||_{L^2(D)}}{(1-\delta)\bar{a}\sqrt{1+C_P^2}}, \qquad \forall p \geq 1$$

Then

$$||u(\mathbf{y}) - u_h(\mathbf{y})||_{L^2(D)} + h||\nabla u(\mathbf{y}) - \nabla u_h(\mathbf{y})||_{L^2(D)} \leq C_u h^2$$

Since the bound is uniform in $\mathbf{y}$*, all moments are bounded*

$$\mathbb{E}\left[||u(\mathbf{y}) - u_h(\mathbf{y})||^p_{L^2(D)}\right]^{\frac{1}{p}} + h\mathbb{E}\left[||\nabla u(\mathbf{y}) - \nabla u_h(\mathbf{y})||^p_{L^2(D)}\right]^{\frac{1}{p}} \leq C_u h^2, \qquad \forall p \geq 1$$

Statistical error

Assuming that Q is a globally Lipschitz functional one can observe that

$$\begin{aligned}\mathscr{E}_h^Q(M) &= \mathbb{E}\left[Q(u_h(\mathbf{y}))\right] - \frac{1}{M}\sum_{m=1}^{M} Q(u_h(\mathbf{y}(\omega_m))) \\ &= \frac{1}{M}\sum_{m=1}^{M}\left[\mathbb{E}\left[Q(u_h(\mathbf{y}))\right] - Q(u_h(\mathbf{y}(\omega_m)))\right]\end{aligned}$$

and, if one takes the expectation with respect to the random sample,

$$\mathbb{E}\left[\mathscr{E}_h^Q(M)\right] = 0$$

Then

$$\mathrm{Var}\left[\mathscr{E}_h^Q(M)\right] = \frac{1}{M}\mathrm{Var}\left[Q(u_h(\mathbf{y}))\right]$$

and one can estimate

$$\begin{aligned}\mathrm{Var}\left[Q(u_h(\mathbf{y}))\right] &\leq \mathbb{E}\left[Q(u_h(\mathbf{y}))^2\right] \\ &\leq C_Q^2||u_h||^2_{L^2_\rho(\Gamma;V)} \\ &\leq 2C_Q^2(||u||^2_{L^2_\rho(\Gamma;V)} + ||u-u_h||^2_{L^2_\rho(\Gamma;V)}) \\ &\leq 2C_Q^2(||u||^2_{L^2_\rho(\Gamma;V)} + h^{2\alpha}||C_u||^2_{L^2_\rho(\Gamma;V)}) < \infty\end{aligned}$$

In the end one can conclude that for Lipschitz functionals Q one has

$$\mathrm{Var}\left[\mathscr{E}_h^Q(M)\right] \leq \frac{C}{M} \to 0 \quad \text{as} \quad M \to \infty$$

with constant C uniformly bounded with respect to h. Moreover, since $\mathrm{Var}\left[Q(u_h(\mathbf{y}))\right] \leq C$ one can apply the law of large numbers, law of iterated logarithms and the Central Limit Theorem. In particular, the previous result implies, via the Chebyshev inequality, convergence in probability, that is, for any given $\varepsilon > 0$

$$P(|\mathscr{E}_h^Q(M)| > \varepsilon) \leq \frac{\mathrm{Var}\left[\mathscr{E}_h^Q(M)\right]}{\varepsilon^2} \leq \frac{C}{M\varepsilon^2} \to 0 \quad \text{as} \quad M \to \infty$$

2.4.2 Complexity analysis

In the end one has that the error can be split in two sources as

$$\mathbb{E}\left[Q(u(\mathbf{y}))\right] - \frac{1}{M}\sum_{m=1}^{M} Q(u_h(\mathbf{y}(\omega_m))) = \mathscr{E}^Q(h) + \mathscr{E}_h^Q(M)$$

where

$$|\mathscr{E}^Q(h)| = |\mathbb{E}\left[Q(u(\mathbf{y})) - Q(u_h(\mathbf{y}))\right]| \leq Ch^\alpha$$

is the discretization error and

$$|\mathscr{E}_h^Q(M)| = |\mathbb{E}\left[Q(u_h(\mathbf{y}))\right] - \frac{1}{M}\sum_{m=1}^{M} Q(u_h(\mathbf{y}(\omega_m)))| \leq C_0\sqrt{\frac{\mathrm{Var}\left[Q(u_h)\right]}{M}}$$

is the statistical error motivated by the Central Limit Theorem, i.e.

$$P(\sqrt{M}|\mathscr{E}_h^Q(M)| \leq c_0\sqrt{\mathrm{Var}\left[Q(u_h)\right]}) \to 2\Phi(c_0) - 1 \quad \text{as} \quad M \to \infty$$

Let be assumed now that the computational work to solve for each $u(\mathbf{y}(\omega_m))$ is $O(h^{-d\gamma})$, where d is the dimension of the computational domain and $\gamma > 0$ represents the complexity of generating one sample with respect to the number of degrees of freedom. One has therefore the following estimates

$$W \propto Mh^{-d\gamma}$$

for the total work and

$$|\mathscr{E}^Q(h)| + |\mathscr{E}_h^Q(M)| \leq C_1h^\alpha + \frac{C_2}{\sqrt{M}}$$

for the total error. What one would like to do is to choose optimally h and M and one possible way is to minimize the computational work subject to an accuracy constraint, i.e. the problem can be written

$$\begin{cases} \text{find } \min_{h,M} Mh^{-d\gamma} \text{ s.t.} \\ C_1h^\alpha + \frac{C_2}{\sqrt{M}} \leq \mathrm{TOL} \end{cases}$$

The Lagrangian of the above problem is

$$\mathscr{L}(M,h,\lambda) = Mh^{-d\gamma} + \lambda(C_1h^\alpha + \frac{C_2}{\sqrt{M}} - \mathrm{TOL})$$

Therefore one gets to the system

$$\begin{cases} \frac{\partial\mathscr{L}}{\partial M} = h^{-d\gamma} - \frac{\lambda C_2}{2\sqrt{M^3}} = 0 \\ \frac{\partial\mathscr{L}}{\partial h} = -d\gamma Mh^{-d\gamma-1} + \lambda C_1\alpha h^{\alpha-1} = 0 \\ \frac{\partial\mathscr{L}}{\partial\lambda} = C_1h^\alpha + \frac{C_2}{\sqrt{M}} - \mathrm{TOL} = 0 \end{cases}$$

Taking in consideration the first and the second equation, resolving with respect to M and eliminating λ one get to

$$\frac{1}{\sqrt{M}} = \frac{2\alpha C_1}{d\gamma C_2} h^\alpha$$

using the third equation and resolving with respect to h one has

$$C_1 h^\alpha + C_2 \frac{2\alpha C_1}{d\gamma C_2} - \text{TOL} = 0$$

$$h^\alpha = \text{TOL} \frac{d\gamma}{C_1(d\gamma + 2\alpha)} \rightarrow \frac{1}{\sqrt{M}} = \text{TOL} \frac{2\alpha}{C_2(d\gamma + 2\alpha)}$$

Since

$$\frac{1}{\sqrt{M}} \propto \text{TOL} \rightarrow M \propto \text{TOL}^{-2}$$

$$h^\alpha \propto \text{TOL} \rightarrow h \propto \text{TOL}^{1/\alpha}$$

the resulting complexity is then

$$W \propto \text{TOL}^{-(2+d\gamma/\alpha)} \tag{2.4}$$

2.4.3 Numerical tests

In this case we want to look at the solution of a one dimensional boundary value problem

$$-(a(x,\omega)u'(x,\omega))' = \underbrace{4\pi^2 \cos(2\pi x)}_{=:f(x)}, \quad \text{for } x \in (0,1)$$

with $u(0,.) = u(1,.) = 0$ and we are interested in computing the expected value of the following QoI

$$Q(u(\omega)) = \int_0^1 u(x,\omega)\,dx$$

We use an uniform $I+1$ uniform grid $0 = x_0 < x_1 < \dot{<} x_I = 1$ on $[0,1]$ with uniform spacing $h = x_i - x_{i-1} = 1/I$, $i = 1,\ldots,I$. Using this grid we can build a piecewise linear FEM approximation,

$$u_h(x,\omega) = \sum_{i=1}^{I-1} \mathbf{u}_{h,i}(\omega)\varphi_i(x),$$

yielding a tridiagonal linear system for the nodal values, $A(\omega)\mathbf{u}_h(\omega) = F$, with

$$A_{i,i-1}(\omega) = -\frac{a(x_{i-1/2},\omega)}{h^2}$$

$$A_{i,i}(\omega) = \frac{a(x_{i-1/2},\omega) + a(x_{i+1/2},\omega)}{h^2}$$

$$A_{i,i+1}(\omega) = -\frac{a(x_{i+1/2},\omega)}{h^2}$$

and

$$F_i = f(x_i).$$

Here we used the notation $x_{i+1/2} = \frac{x_i+x_{i+1}}{2}$ and $x_{i-1/2} = \frac{x_i+x_{i-1}}{2}$. The integral in $Q(u_h)$ can be then computed exactly by a trapezoidal method, yielding

$$Q(u_h) = h\sum_{i=1}^{I-1} \mathbf{u}_{h,i}.$$

We look at two different models for the diffusion coefficient $a(x,\omega)$. The first is a *piecewise constant* model and we define it as

$$a(x,\omega) = 1 + \sigma \sum_{n=1}^{N} Y_n(\omega)\mathbb{I}_{[\hat{x}_{n-1},\hat{x}_n]}(x),$$

with equispaced nodes $\hat{x}_n = \frac{n}{N}$ for $0 \le n \le N$ and i.i.d. uniform random variables $Y_n \sim U([-\sqrt{3},\sqrt{3}])$. Tests are conducted for different uniform mesh refinements, i.e. values of $I = N2^l$, $l \ge 0$. We also try different number of input random variables $N = 10$, $N = 20$ and $N = 40$. The constant σ has been chosen ensuring coercivity, namely $1 - \sigma\sqrt{3} > 0$.

The second model is defined as

$$a(x,\omega) = \exp(\kappa(x,\omega))$$

where $\kappa(x,\omega)$ is a stationary random field with the Matérn covariance function

$$C(x,y) = \sigma^2 \frac{1}{\Gamma(\nu)2^{\nu-1}}\Big(\sqrt{2\nu}\frac{|x-y|}{\rho}\Big)^{\nu} K_\nu\Big(\sqrt{2\nu}\frac{|x-y|}{\rho}\Big)$$

where Γ is the gamma function and K_ν is the modified Bessel function of the second kind. We look at the following special cases of C with $\rho = 0.1$ and $\sigma^2 = 2$

$$\begin{aligned}
\nu = 0.5, \quad C(x,y) &= \sigma^2 \exp\Big(-\frac{|x-y|}{\rho}\Big)\\
\nu = 1.5, \quad C(x,y) &= \sigma^2\Big(1 + \frac{\sqrt{3}|x-y|}{\rho}\Big)\exp\Big(-\frac{\sqrt{3}|x-y|}{\rho}\Big)\\
\nu = 2.5, \quad C(x,y) &= \sigma^2\Big(1 + \frac{\sqrt{5}|x-y|}{\rho} + \frac{\sqrt{3}|x-y|^2}{\rho^2}\Big)\exp\Big(-\frac{\sqrt{5}|x-y|}{\rho}\Big)\\
\nu \to \infty, \quad C(x,y) &= \sigma^2 \exp\Big(-\frac{|x-y|^2}{2\rho^2}\Big)
\end{aligned}$$

For this example we chose to represent the stationary random $\kappa(x,\omega)$ using the truncated Karhunen-Loève expansion with N terms

$$\kappa(x,\omega) \approx \sum_{n=1}^{N} \sqrt{\lambda_n} Y_n(\omega) e_n(x)$$

where $\{Y_k\}$ is a set of i.i.d. uniform random variables over $[-\sqrt{3}, \sqrt{3}]$. To find the eigenfunction and eigenvalues of C we solve the eigenvalue problem

$$\int_0^1 C(x,y)e_n(y)dy = \lambda_n e_n(x) \tag{2.5}$$

we do this by discretizing C as a matrix by evaluating the function $C(x_{i+\frac{1}{2}}, x_{j+\frac{1}{2}})$ over the grid $\{x_{i+\frac{1}{2}}\}_{i=0}^{I-1} \times \{x_{i+\frac{1}{2}}\}_{i=0}^{I-1}$ with $N \leq I$. Then we use `MATLAB`'s function `eig()` and we place the eigenvector in a decreasing order, namely

$$\lambda_1 < \lambda_2 < \cdots < \lambda_N$$

This discretization corresponds to a piecewise constant FEM approximation to the eigenvalue problem (2.5).

With the theory that lies behind the Karhunen-Loève expansion we could fill an entire section. In this work we decided to avoid the explanations as we used it as a tool to develop our calculations.

In Figure 2.5 we present the results related to Model 1. We decide to represent the expected value of the solution (black line) in each point of the domain with the relative standard deviation (red lines). We do that for different mesh refinements (left column) and for a different number of random variables (right column), namely in the left column the refinements are shown while in the right column the increasing random variables. The main result that we want to point out is the *homogenization* of the random coefficient. This phenomenon consists in a sort of averaging of the coefficient which can be approximated as a constant one better and better as the number of random variables increases, that is as become more and more oscillating. It can be seen in the significant reduction of variance in the Figure 2.5(f). The conclusion is that the estimate of our quantity of interest gets more and more accurate as the variability of the problem increases.

In Figure 2.6 instead we present the results related to Model 2. Again, in black we represent the expected value of the solution in each point of the domain with the relative standard deviation in red. In the left column we decided to test different truncations of the KL expansion while keeping constant the mesh refinements. In the right column instead we present the results with a different number of samples making sure to choose a number of terms of the KL expansion that covers the 90% of the spectrum. In this case, unlike the previous case, we can see that the more terms we include the more the variability of the problem increases until we cover a reasonable part of the spectrum. On the other hand, if we increase the number of samples, we can see the convergence of the standard deviation and, therefore, of the variance.

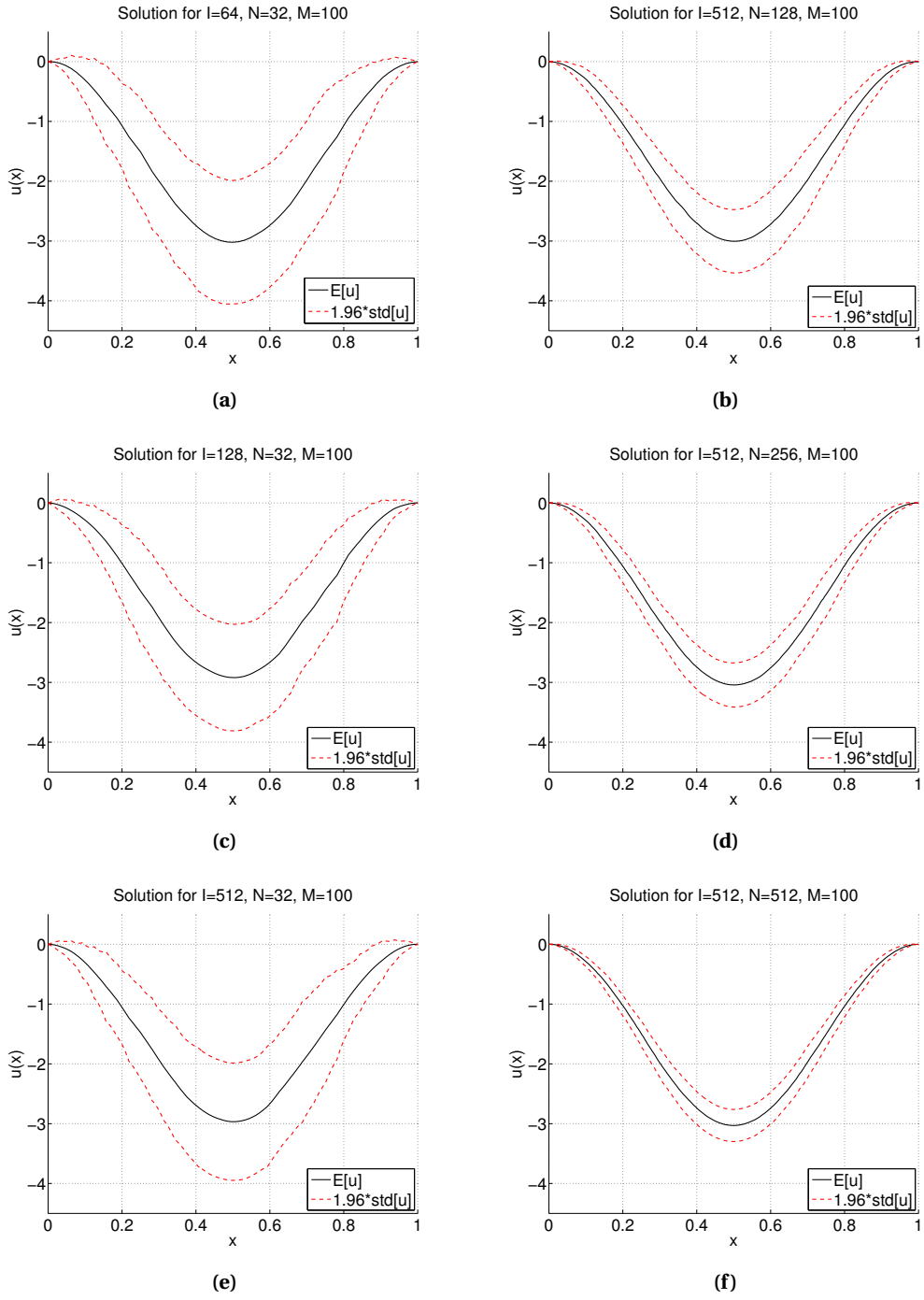


Figure 2.5: Numerical simulation for Model 1 - Mesh refinement (left column, $I = 1/h = 64, 128, 512$) and increase of the variability (right column $N = 128, 256, 512$). The phenomenon of *homogenization* can be seen very well in the second column.

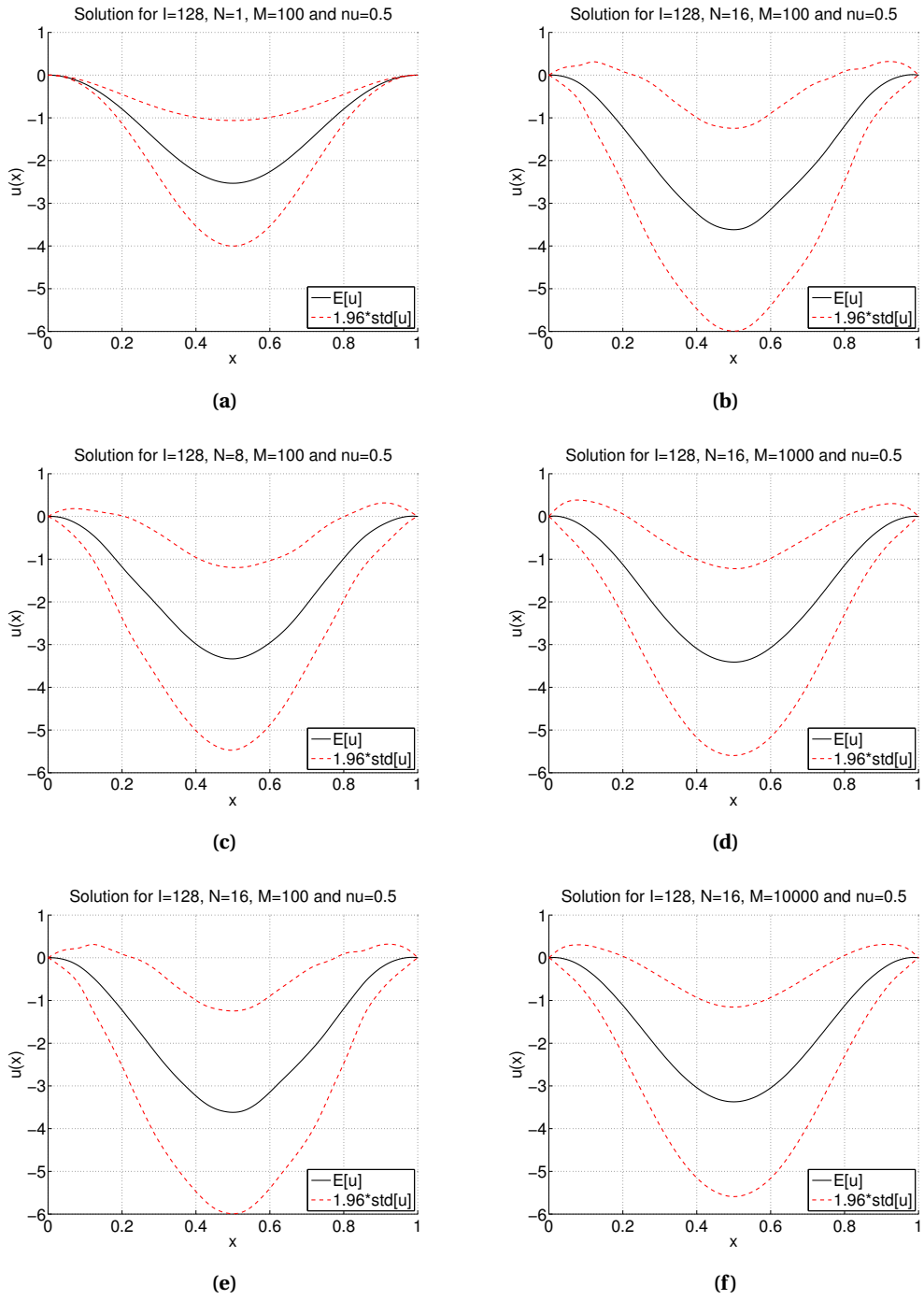


Figure 2.6: Numerical simulation for Model 2 - Different numbers of random variables (left column $N = 1, 4, 8$) and different number of samples (right column $M = 10^2, 10^3, 10^4$).

2.5 Variance Reduction

Since the Monte Carlo error can be written as

$$\mathbb{E}[Y] - \frac{1}{M}\sum_{j=1}^{M} Y(\omega_j) \approx C_\alpha \sqrt{\frac{\mathrm{Var}[Y]}{M}}$$

the paths to reduce it that we can walk through are two: the first is to increase the samples M while the other is reduce the quantity $\mathrm{Var}[Y]$. So the main idea of variance reduction is try to "reduce" $\mathrm{Var}[Y]$ without changing $\mathbb{E}[Y]$. In practice, one wants to find another RV Z such that:

- $\mathbb{E}[Z] = \mathbb{E}[Y]$;
- $\mathrm{Var}[Z] \ll \mathrm{Var}[Y]$;

and then apply the Monte Carlo method to the variable Z instead of Y in a way that

$$\mathbb{E}[Y] - \frac{1}{M}\sum_{j=1}^{M} Z(\omega_j) \approx C_\alpha \sqrt{\frac{\mathrm{Var}[Z]}{M}} \ll C_\alpha \sqrt{\frac{\mathrm{Var}[Y]}{M}}.$$

2.5.1 Control Variate

In this case the idea is to look for a RV X that has a strong correlation (positive or negative) with Y and a known mean $\mathbb{E}[X]$, generate a sample of both RVs and combine the empirical means to an estimator with lower variance than the MC one.

Then, one can define a new random variable

$$Z(\beta) = Y - \beta(X - \mathbb{E}[X])$$

The point is that since since $\mathbb{E}[X]$ is known, we are free to add a term $\beta(X - \mathbb{E}[X])$ with mean zero to the MC estimator, so that the unbiasedness is preserved. The variance is:

$$\begin{aligned}
\mathrm{Var}\left[Z(\beta)\right] &= \mathbb{E}\left[(Y - \beta(X - \mathbb{E}[X]) - \mathbb{E}[Y])^2\right] \\
&= \mathbb{E}\left[((Y - \mathbb{E}[Y]) - \beta(X - \mathbb{E}[X]))^2\right] \\
&= \mathrm{Var}[Y] + \beta^2 \mathrm{Var}[X] - 2\beta \,\mathrm{Cov}(X, Y)
\end{aligned}$$

If one derives this last expression with respect to β, the minimum can be easily found to be

$$\beta^* = \frac{\mathrm{Cov}(X, Y)}{\mathrm{Var}[X]}$$

and one has a total variance reduction of

$$\text{Var}\left[Z(\beta^*)\right] = \text{Var}\left[Y\right]\left(1 - \frac{(\text{Cov}(X,Y))^2}{\text{Var}\left[X\right]\text{Var}\left[Y\right]}\right) < \text{Var}\left[Y\right]$$

In practice, all the quantities are approximate using sample covariances and variances. It should be pointed out that this kind of variance reduction really pays off under certain conditions, i.e. if the assumption that the work to generate the pair (X_j, Y_j) is $(1+\theta)$ times the work to generate Y_j is made one has that this strategy is convenient only if

$$\text{Var}\left[Y\right] > (1+\theta)\,\text{Var}\left[V(\beta^*)\right]$$

then, we can remark the previous condition as

$$1 > (1+\theta)(1-\rho^2) \quad \rightarrow \quad \rho^2 > \frac{1}{1+\theta}$$

Example *Consider Monte Carlo integration for calculating* $I = \int_0^1 f(x)dx$. *This integral can be seen as the expected value of* $f(U)$, *where*

$$f(x) = \frac{1}{1+x}$$

and U follows a uniform distribution [0,1]. Using a sample of size M and denoting the sample as $u_1, \ldots, u_M$, *one has that the estimate is given by*

$$I \approx \frac{1}{M}\sum_{i=1}^{M} f(u_i).$$

If one introduces $g(x) = 1 + x$ *as a control variate with a known expected value*

$$\mathbb{E}\left[g(U)\right] = \int_0^1 (1+x)dx = \frac{3}{2}$$

can combine the two into a new estimate

$$I \approx \frac{1}{M}\sum_{i=1}^{M} f(u_i) - \hat{\beta}^*\left(\frac{1}{M}\sum_{i=1}^{M} g(u_i) - \frac{3}{2}\right)$$

Using $M = 5000$, *the following results has been obtained, where the coefficient* $\hat{\beta}^*$ *has been esti-*

	Estimate	Variance
Classical estimate	0.6952	0.0196
Control variate	0.6931	0.0006

mated with the sample covariance and variance respectively

$$s_{f,g} = \frac{1}{M-1}\sum_{j=1}^{M}(f(u_j) - \frac{1}{M}\sum_{i=1}^{M} f(u_i))(g(u_j) - \frac{1}{M}\sum_{i=1}^{M} g(u_i))$$

$$s_g = \frac{1}{M-1}\sum_{j=1}^{M}(g(u_j) - \frac{1}{M}\sum_{i=1}^{M} g(u_i))^2$$

so that $\hat{\beta}^* \approx 0.4768$ *and the variance has been reduced by* 97%. *Note that the exact result is* $\ln 2 \approx 0.69314718$

Although this example is very simple and intuitive, a few words must be spent on the kind of control variables are typically used in practice. In theory, any variable X correlated with Y and whose expectation is known can be used as a control variable. This means that the potential candidates to be used are quantities that are closely related to the one for which we try to estimate the mean, but that are, in some sense, simpler and whose expectation is known.

In the context of SPDE or PDE with random coefficient we can apply this idea using a coarser discretization of the problem (e.g. with double the gridsize h) with the same realization of $\{Y_k\}$ as a control variable for the approximation of $E[Q(u_h)]$. This idea is the basis of multilevel Monte Carlo and the results will be shown in the next chapter.

CHAPTER 3

Multilevel Monte Carlo Method

3.1 Introduction

Monte Carlo methods are a very general and useful approach to the estimation of quantities arising from stochastic simulation. However they can be computationally expensive, particularly when the cost of generating individual stochastic samples is very high, as in the case of stochastic PDEs. Multilevel Monte Carlo is a recently developed approach which greatly reduces the computational cost by performing most simulations with low accuracy at a correspondingly low cost, with relatively few simulations being performed at high accuracy and a high cost.

According to [16], when the dimensionality of the uncertainty (or the uncertain input parameters) is low, it can be appropriately modeled using the Fokker-Planck PDE (when is the Eulerian counterpart of an SDE) and using stochastic Galerkin, stochastic collocation or polynomial chaos methods (Xiu and Karniadakis 2002, Babuška, Tempone and Zouraris 2004, Babuška, Nobile and Tempone 2010, Gunzburger, Webster and Zhang 2014). When the level of uncertainty is low, and its effect is largely linear, then moment methods can be an efficient and accurate way in which to quantify the effects on uncertainty (Putko, Taylor, Newman and Green 2002). However, when the uncertainty is high-dimensional and strongly non-linear, Monte Carlo simulation remains the preferred approach.

Monte Carlo simulation is extremely simple as we could appreciate in the previous chapters and we already know that its weakness relies in its computational cost that can be very high,

particularly when when each sample might require the approximate solution of a PDE, or a computation with many time steps.

Giles and Waterhouse in [17] developed a variant of MLMC which uses *quasi-Monte Carlo* samples instead of independent Monte Carlo samples. The numerical results were very encouraging for SDE applications in which the dominant computational cost was on the coarsest levels of resolution where QMC is known to be most effective due to the low number of dimensions involved. However there was no supporting theory for this research.

More recently, there has been considerable research on the theoretical foundations for multilevel QMC (MLQMC) methods (Niu, Hickernell, Müller-Gronbach and Ritter 2010, Kuo, Schwab and Sloan 2012, Dick et al. 2013, Baldeaux and Gnewuch 2014, Dick and Gnewuch 2014a, Dick and Gnewuch 2014b). These theoretical developments are very encouraging, and may lead to the development of new, more efficient, multilevel methods.

3.2 Multilevel Monte Carlo method

The history of multilevel Monte Carlo can be traced back to Heinrich et al. [21, 22], where it was introduced in the context of parametric integration. Kebaier [23] then used similar ideas for a two-level Monte Carlo method to approximate weak solutions to stochastic differential equations in mathematical finance. As we have already seen, one of the classic approaches to Monte Carlo variance reduction is through the use of a control variate which is well correlated with the variable we want to approximate and has a known expectation. In [15], Giles extended this idea to more that two levels and dubbed his extension the Multilevel Monte Carlo (MLMC) method.

The multilevel idea comes from the control variate technique where, instead of finding another random variable correlated to the first one, one use the same random variable but with a different approximation so the sampling is made not just from one approximation of a certain quantity of interest, but from several.

Let g denote a functional of a solution u of an underlying stochastic model and let g_l denote the corresponding approximation on level l corresponding to a discretization $h_l \in \{h_l\}_{l=0}^{L}$.

The goal, as for the Monte Carlo method, is to estimate $\mathbb{E}\left[g\right]$ while controlling the error that will be defined below. The expected value of the finest approximation can be expressed as

$$\mathbb{E}\left[g_L\right] = \mathbb{E}\left[g_0\right] + \sum_{l=1}^{L} \mathbb{E}\left[g_l - g_{l-1}\right]$$

That it can be seen as a control variate if one develops it as a telescopic series

$$\begin{aligned}\mathbb{E}\left[g_L\right] &= \left(\mathbb{E}\left[g_L\right]-\mathbb{E}\left[g_{L-1}\right]\right)+\mathbb{E}\left[g_{L-1}\right]\\ &= \left(\mathbb{E}\left[g_L\right]-\mathbb{E}\left[g_{L-1}\right]\right)+\left(\mathbb{E}\left[g_{L-1}\right]-\mathbb{E}\left[g_{L-2}\right]\right)+\mathbb{E}\left[g_{L-2}\right]\\ &= \ldots\end{aligned}$$

where each variable is controlled by itself at a lower level. Then the MLMC estimator is

$$Y=\sum_{l=0}^{L} Y_l, \qquad Y_l = M_l^{-1}\sum_{m=1}^{M_l}\left(g_l(\omega_{l,m})-g_{l-1}(\omega_{l,m})\right) \tag{3.1}$$

with $g_{-1}\equiv 0$ and

$$\mathbb{E}[Y]=\mathbb{E}\left[g_L\right], \qquad \mathrm{Var}[Y]=\sum_{l=0}^{L} M_l^{-1}V_l, \qquad V_l\equiv \mathrm{Var}\left[g_l-g_{l-1}\right]$$

It is important to emphatize that the quantity $g_l(\omega_{l,m})-g_{l-1}(\omega_{l,m})$ in (4) comes from using the same random sample on both levels.

If we define C_0, V_0 to be the cost and variance of one sample of g_0, and C_l, V_l to be the cost and variance of one sample of $P_l - P_{l-1}$, then the overall cost and variance of the multilevel estimator is $\sum_{l=0}^{L} M_l C_l$ and $\sum_{l=0}^{L} M_l^{-1} V_l$ respectively.

For a fixed cost, the variance is minimized by choosing M_l to minimize

$$\sum_{l=0}^{L}\left(M_l C_l + \mu^2 M_l^{-1} V_l\right)$$

for some value of the Lagrange multiplier μ^2. This gives $M_l = \mu\sqrt{V_l/C_l}$. To achieve an overall variance of ε^2 then it is required that $\mu = \varepsilon^{-2}\sum_{l=0}^{L}\sqrt{V_l C_l}$, and the total computational cost is then

$$C=\varepsilon^{-2}\Big(\sum_{l=0}^{L}\sqrt{V_l C_l}\Big)^2.$$

It is important to note whether the product $V_l C_l$ increases or decreases with l, i.e. whether or not the cost increases with level faster than the variance decreases. If it increases with level, so that the dominant contribution to the cost comes from $V_L C_L$ then we have $C \approx \varepsilon^{-2} V_L C_L$, whereas if it decreases and the dominant contribution comes from $V_0 C_0$ then $C \approx \varepsilon^{-2} V_0 C_0$. This contrast to the standard MC cost of approximately $\varepsilon^{-2} V_0 C_L$, assuming that the cost of computing g_L is similar to the cost of computing $g_L - g_{L-1}$ and that $\mathrm{Var}\left[g_L\right] \approx \mathrm{Var}\left[g_0\right]$.

This shows that in the first case the MLMC cost is reduced by a factor V_L/V_0, corresponding to the ratio of the variances $\mathrm{Var}\left[g_L - g_{L-1}\right]$ and $\mathrm{Var}\left[g_0\right]$, whereas in the second case it is

reduced by a factor C_0/C_L, the ratio of the cost of computing g_0 and $g_L - g_{L-1}$. If the product $V_l C_l$ does not vary with level, then the total cost is $\varepsilon^{-2} L^2 V_0 C_0 = \varepsilon^{-2} L^2 V_L C_L$.

Since the multilevel estimator Y is an approximation of $\mathbb{E}\left[g\right]$, then, according to [9], we can define the mean square error (MSE) as

$$\text{MSE} \equiv \mathbb{E}\left[(Y - \mathbb{E}\left[g\right])^2\right] = \text{Var}\left[Y\right] + (\mathbb{E}\left[Y\right] - \mathbb{E}\left[g\right])^2 \tag{3.2}$$

To ensure that the MSE is less than ε^2, it is sufficient to ensure that $(\mathbb{E}\left[g_L - g\right])^2$ and $\text{Var}\left[Y\right]$ are both less than $\frac{1}{2}\varepsilon^2$. Combining this idea with a geometric sequence of levels in which the cost increases exponentially with level, while both the weak error $\mathbb{E}\left[g_L - g\right]$ and the multilevel correction variance V_l decrease exponentially, we can state the following theorem

Theorem 4 (Cliffe, Giles, Scheichl, Teckentrup)**.** *Let g denote a random variable and let g_l denote the corresponding level l numerical approximation.*

If there exist independent estimators Y_l based on M_l Monte Carlo samples, each with expected cost C_l and variance V_l, and positive constants $\alpha, \beta, \gamma, c_1, c_2, c_3$ such that $\alpha \geq \frac{1}{2}\min(\beta, \gamma)$ and

1. $|\mathbb{E}\left[g_l - g\right]| \leq c_1 2^{-\alpha l}$
2. $\mathbb{E}\left[Y_l\right] = \begin{cases} \mathbb{E}\left[g_0\right], & l = 0 \\ \mathbb{E}\left[g_l - g_{l-1}\right], & l > 0 \end{cases}$
3. $V_l \leq c_2 2^{-\beta l}$
4. $C_l \leq c_3 2^{\gamma l}$

then there exists a positive constant c_4 such that for any $\varepsilon < e^{-1}$ there are values L and M_l for which the multilevel estimator

$$Y = \sum_{l=0}^{L} Y_l$$

has a mean-square-error with bound

$$MSE \equiv \mathbb{E}\left[(Y - \mathbb{E}\left[g\right])^2\right] < \varepsilon^2$$

with a computational complexity C with bound

$$\mathbb{E}\left[C\right] = \begin{cases} c_4 \varepsilon^{-2}, & \beta > \gamma, \\ c_4 \varepsilon^{-2} \log(\varepsilon)^2, & \beta = \gamma, \\ c_4 \varepsilon^{-2-(\gamma-\beta)/\alpha}, & \beta < \gamma. \end{cases}$$

The statement of the theorem is a slight generalization of the original theorem in [Giles,2008b]. It corresponds to the theorem and proof in [9], except for the minor change to the expected costs to allow for applications in which the simulation cost of individual samples is itself random. Note that if condition 3. is tightened slightly to be a bound on $\mathbb{E}\left[(g_l - g_{l-1})^2\right]$, that is usually the quantity which is bounded in numerical analysis, then it would follow that $\alpha \geq \frac{1}{2}\beta$.

From the theorem above we can make the following assertions.

In the case $\beta > \gamma$, the dominant computational cost is on the coarsest levels where $C_l = O(1)$ and $O(\varepsilon^{-2})$ samples are required to achieve the desired accuracy. This is the standard result for a Monte Carlo approach using i.i.d. samples; to do better would require an alternative approach such as the use of Latin hypercube or quasi-Monte Carlo methods.

In the case $\beta < \gamma$, the dominant computational cost is on the finest levels. Because of condition 1., $2^{-\alpha L} = O(\varepsilon)$ and hence $C_L = O(\varepsilon^{-\gamma/\alpha})$. If $\beta = 2\alpha$, which is usually the best that can be achieved since typically $\mathrm{Var}\left[g_l - g_{l-1}\right]$ is similar in magnitude to $\mathbb{E}\left[(g_l - g_{l-1})^2\right]$ which is greater than $(\mathbb{E}\left[g_l - g_{l-1}\right])^2$, than the total cost is $O(C_L)$, corresponding to $O(1)$ samples on the finest level which is the best that can be achieved.

The last case of $\beta = \gamma$ is the one for which both the computational effort and the contributions to the overall variance are spread approximately evenly across all the level.

3.2.1 MLMC implementation

Based on the theory in the previous section, the geometric MLMC algorithm used for the numerical test is:

Algorithm 3 MLMC

1: start with $L = 2$ and initial target of M_0 samples on levels $l = 0, 1, 2$
2: **while** extra samples need to be evaluated **do**
3: evaluate extra samples on each level
4: compute/update estimates on each level for V_l, $l = 0, \ldots, L$
5: define optimal M_l, $l = 0, \ldots, L$
6: test for weak convergence
7: if not converged, set $L := L + 1$ and initialize target M_L
8: **end while**

In the above algorithm, the equation for the optimal M_l is

$$M_l = \left\lceil 2\varepsilon^{-2}\sqrt{V_l/C_l}\left(\sum_{l=0}^{L}\sqrt{V_l C_l}\right)\right\rceil$$

where V_l is the estimated variance and C_l is the cost of an individual sample on level l. This ensures that the estimated variance of the combined multilevel estimator is less than $\frac{1}{2}\varepsilon^2$.

The test for weak convergence tries to ensure that $|\mathbb{E}\left[g - g_L\right]| < \varepsilon/\sqrt{2}$, to achieve an MSE which is less than ε^2, with ε being a user-specified r.m.s. accuracy. If $|\mathbb{E}\left[g - g_L\right]| \propto 2^{-\alpha l}$ then the remaining error is

$$\mathbb{E}\left[g - g_L\right] = \sum_{l=L+1}^{\infty} \mathbb{E}\left[g_l - g_{l-1}\right] = \frac{\mathbb{E}\left[g_L - g_{L-1}\right]}{(2^\alpha - 1)}$$

This leads to the convergence test $|\mathbb{E}\left[g_L - g_{L-1}\right]|/(2^\alpha - 1) < \varepsilon/\sqrt{2}$, but for sake of robustness, we extend this check to extrapolate from the previous two data points $|\mathbb{E}\left[g_{L-1} - g_{L-2}\right]|$, $|\mathbb{E}\left[g_{L-2} - g_{L-3}\right]|$, and take the maximum over all three as the estimated remaining error. This is possible for very simple problem, like the first we are presenting in the next section. As the problem grows in complexity, the task becomes harder and harder.

The results presented later use the two routines given by [16] written in MATLAB:

- `mlmc.m`: driver code which performs the MLMC calculation using an application-specific routine to compute $\sum_n (g_l^{(n)} - g_{l-1}^{(n)})^p$ for $p = 1, 2$ and a specified number of independent samples;
- `mlmc_test.m`: a program which perform various tests and then calls `mlmc.m` to perform a number of MLMC calculations.

In the results are also reported the consistency check versus level plot and the kurtosis versus level plot. These need some explanation. If a, b, c are estimates for $\mathbb{E}\left[g_{l-1}^f\right]$, $\mathbb{E}\left[g_l^f\right]$, $\mathbb{E}\left[Y_l\right]$, respectively, then it should be true that $a - b + c \approx 0$. The consistency check verifies that this is true.

Since

$$\sqrt{\mathrm{Var}\left[a - b + c\right]} \leq \sqrt{\mathrm{Var}\left[a\right]} + \sqrt{\mathrm{Var}\left[b\right]} + \sqrt{\mathrm{Var}\left[c\right]}$$

it computes and plots the ratio

$$\frac{|a - b + c|}{3(\sqrt{V_a} + \sqrt{V_b} + \sqrt{V_c})} \tag{3.3}$$

where V_a, V_b, V_c are empirical estimates for the variances of a, b, c. The probability of this ratio being greater than unity is less than 0.3%. Hence, if so, it indicates a likely programming error or a mathematical error in the formulation which violates the crucial identity

$$\mathbb{E}\left[g_l^f\right] = \mathbb{E}\left[g_l^c\right]$$

that provides the same expectation on level l for the two approximations[1].

[1] according to Giles the superscript f stands for "fine" while c stands for "coarse".

The MLMC approach needs a good estimate for $V_l = \mathrm{Var}[Y_l]$, to achieve that at least 10 samples are needed and they may be sufficient in many cases for a rough estimate, but many more are needed when there are rare outliers. When the number of samples M is large, the standard deviation of the sample variance for a random variable X with zero mean is approximately $\sqrt{(\kappa-1)/M}\mathbb{E}\left[X^2\right]$ where the kurtosis κ is defined as $\kappa = (\mathbb{E}\left[X^4\right]/\mathbb{E}\left[X^2\right])^2$. If this quantity is very large it could indicate that the empirical variance estimate is poor.

3.3 Numerical tests

In this section we construct a Multilevel Monte Carlo estimate for the quantity $\mathbb{E}[Q(u)]$ for the boundary-value problem already seen in the Standard Monte Carlo chapter.

We recall that the problem is

$$-(a(x,\omega)u'(x,\omega))' = \underbrace{4\pi^2\cos(2\pi x)}_{=:f(x)}, \quad \text{for } x \in (0,1)$$

with $u(0,.) = u(1,.) = 0$ and we are interested in computing the expected value of the following QoI

$$g(u(\omega)) = \int_0^1 u(x,\omega)\,dx$$

The diffusion term is modelled as a piecewise constant coefficient

$$a(x,\omega) = 1 + \sigma \sum_{n=1}^{N} Y_n(\omega)\mathbb{I}_{[\hat{x}_{n-1},\hat{x}_n]}(x),$$

with σ chosen ensuring the coercivity and so equal to $1/2$. Level l uses a uniform grid with spacing $h_l = 2^{-(l+1)}$. A first order finite element approximation is used (piecewise linear finite elements).

The uniform first order accuracy means that there is a constant K such that

$$|g - g_l| < K h_l^2 \tag{3.4}$$

and therefore we should obtain $\alpha = 2$, $\beta = 4$ and $\gamma = 1$, resulting in an $O(\varepsilon^{-2})$ complexity (see Theorem 4). All these features can be verified in the numerical results in Figure 3.1, in Table 3.1 and in Table 3.2.

In Figure 3.1, we have the performance plots for the problem and the quantity of interest described above. In the top row we have the weak and the strong error from which we compute the α and the β of the Theorem 4. As we see in the plot on the left, for large values of h_l, the

variance of g_l and $g_l - g_{l-1}$ are close. Increasing h_l even further, the two graph will eventually cross, and Var$\left[g_l - g_{l-1}\right]$ will be larger than Var$\left[g_l\right]$. In this situation the cost of the MLMC method from level l will actually be bigger than those using standard MC, rendering any further coarsening useless. In the middle row plot we report the consistency check and the kurtosis, the two quantities defined above.

Estimates of the parameters based on linear regression:	
$\alpha = 2.03$	exponent for MLMC weak convergence
$\beta = 4.08$	exponent for MLMC variance
$\gamma = 1.01$	exponent for MLMC cost

Table 3.1: Numerical result for the 1D elliptic PDE with random diffusion solved modeled with a piecewise constant coefficient.

The bottom two plots are related to the implementation of the MLMC algorithm and to its cost. The left plot shows a comparison of the cost of standard MC with the cost of MLMC (the actual results are reported exactly in Table 3.2). Note that the MLMC algorithm does not only result in large savings in the computational cost, but that the cost of the MLMC estimator also grows more slowly then the cost of the standard MC estimator as $\varepsilon \to 0$.

ε	MLMC cost	StdMC cost	Gain factor
0.005	$4.6 \cdot 10^6$	$1.1 \cdot 10^8$	24
0.01	$9.7 \cdot 10^5$	$1.4 \cdot 10^7$	14
0.02	$2.3 \cdot 10^5$	$1.6 \cdot 10^6$	7
0.05	$4.1 \cdot 10^4$	$2.6 \cdot 10^5$	6
0.1	$9.8 \cdot 10^3$	$3.1 \cdot 10^4$	3

Table 3.2: Cost comparison between MLMC and StdMC as $\varepsilon \to 0$ for the 1D elliptic PDE with random diffusion.

From the original MATLAB code provided by [16] we modify only the application-specific routine.

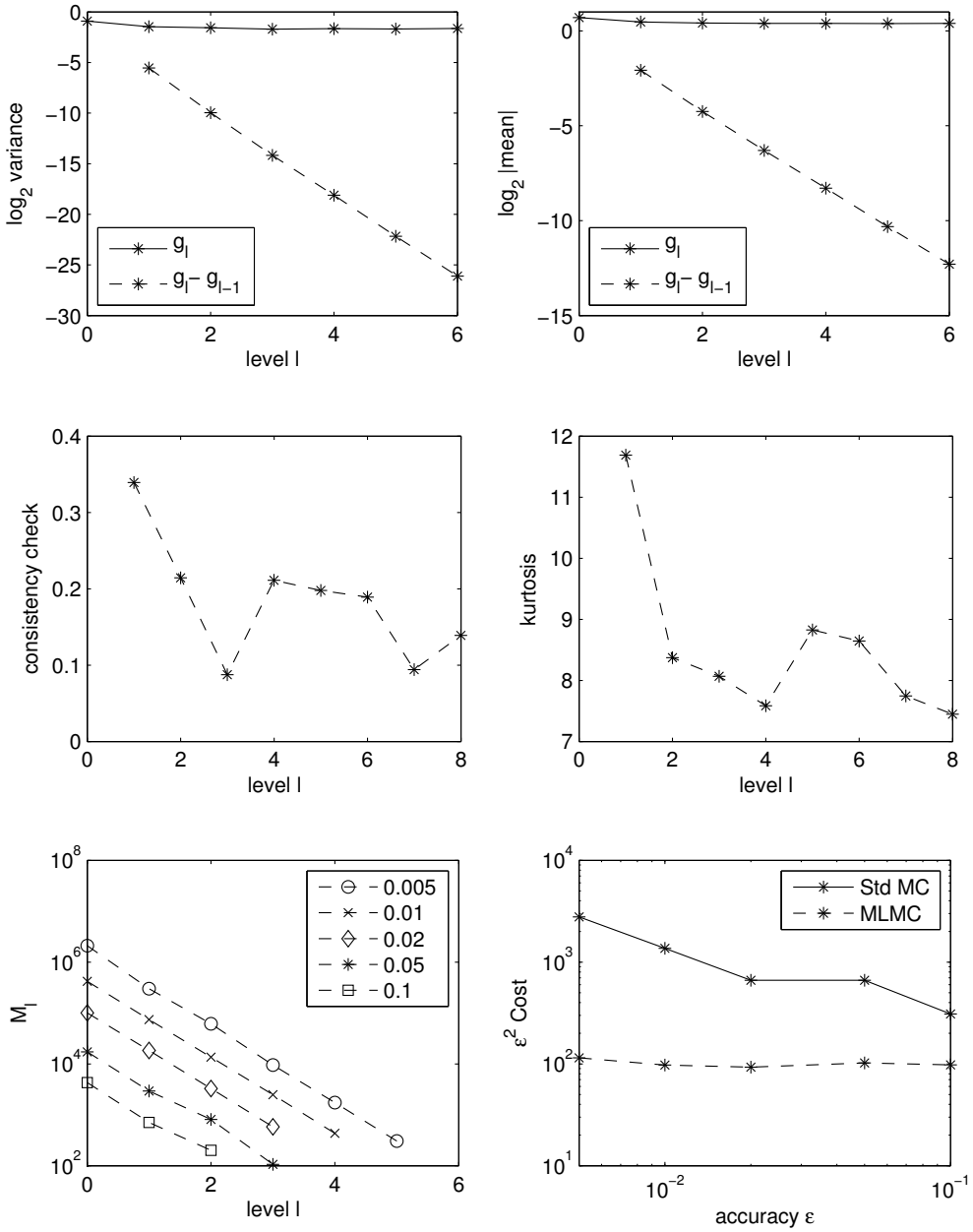


Figure 3.1: Numerical result for the 1D elliptic PDE with random diffusion solved modeled with a piecewise constant coefficient. Rates of strong and weak error (up-left and up-right plots resp.), consistency check (Eq. 3.3) and kurtosis (middle-left and middle-right plots resp.), accuracy vs. cost (low-right) and sample vs. level for different accuracies (low-left).

3.4 Application to random geometry problems

The objective of this study is to characterize the uncertainty in the estimation of parameters, such as permeability, dispersivity and capillary pressure, when they are computed numerically from fluid dynamics simulations at the pore-scale. This is an important aspects that is often overlooked when samples of porous media are studied (either experimentally or numerically). In fact, realistic pore scale simulations can contain a large number of uncertainties, due to the computational and experimental capabilities[2] and to the lack of knowledge in the micro-scale physical parameters, boundary conditions, and geometrical details of the pore space. This can result in a huge variability of estimated upscaled parameters.

Due to the presence of very complex stochastic inputs (such as the geometry), it is very difficult to parametrize explicitly the randomness in terms of a finite number of independent random variables. Therefore it is hard to use standard uncertainty quantification approaches. This problem is instead perfect to test the Multilevel Monte method described previously. However, even if the original algorithm is adaptive, for some cases, the number of levels and the initial number samples is chosen *a priori*.

In a first place we implement the Giles' multilevel algorithm in a new code described accurately in Appendix A and we build some example without any physical relevance in order to test it. Then we construct an appropriate fluid model for the physical phenomenon of interest that in our case will be the pore-scale Navier-Stokes description, then we implement efficiently a numerical approximation based on the finite volume solver `OpenFOAM` [33], and finally we construct an efficient sampling algorithm to estimate the statistics of upscaled parameters.

3.4.1 Heterogeneous materials

Aiming to introduce the problem that we are going to study, we now give some general definitions regarding the physical context. In particular, in the following, we state some considerations about the so-called heterogeneous materials.

According to [37], an *heterogeneous material* is composed of domain of different materials or the same in different states. it is assumed that the "microscopic" length scale is much larger than the molecular dimensions but much smaller than the characteristic length of the macroscopic sample. In such circumstances, the heterogeneous material can be viewed as a continuum on the microscopic scale, and macroscopic or *effective* properties can be ascribed

[2] That, for example, limit the size of the domain to be studied or that are invasive changing the properties of the sample

to it.

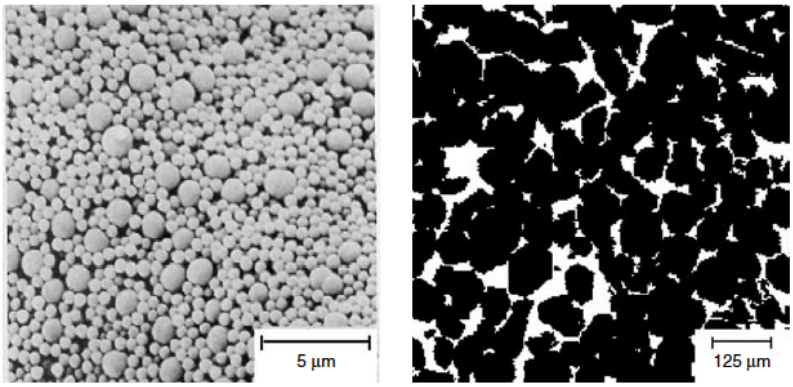


Figure 3.2: Examples of random heterogeneous materials. Left panel: A colloidal system of hard spheres of two different sizes. Right panel: A Fontainebleau sandstone. Images from [37].

The physical phenomena of interest occur on "microscopic" length scales and structures on this scale are generally referred to as "microstructures". In many instances, the microstructures can be characterized only statistically, and therefore such materials are referred to as *random heterogeneous materials*. There is a vast family of random microstructures that are possible and *porous media* are included in it. Figure 3.2 shows examples of synthetic and natural random heterogeneous materials. The first example shows a scanning electron micrograph of a colloidal system of hard spheres of two different sizes, primarily composed of boron carbide (black regions) and aluminum (white regions). The second example shows a planar section through a Fontainebleau sandstone obtained via X-ray microtomography.

There are different classes of problems (we give a general idea in Table 3.3), but here the focus will be put on steady-state effective property associated with the *fluid permeability tensor*, **k**. The knowledge of this effective property is required for many applications in engineering, physics, geology and materials science. This quantity is a key macroscopic property for describing slow viscous flow through porous media and it is the proportionality constant between the average fluid velocity and applied pressure gradient in the porous medium. This relation is Darcy's law for the porous medium and it is known as

$$q = -\frac{\mathbf{k}}{\mu}\nabla P$$

where μ is the dynamic viscosity, q is the velocity field and ∇P is the applied pressure gradient.

The effective properties of a heterogenous material depend on the phase properties and on the microstructural information, including the phase volume fraction which represent the

Class	General effective property (K_e)	Average (or applied) generalized intensity (G)	Average generalized flux (F)
A	Thermal conductivity	Temperature gradient	Heat flux
	Diffusion coefficient	Concentration gradient	Mass flux
B	Elastic moduli	Strain field	Stress field
C	Survival time	Species production rate	Concentration field
D	Fluid permeability	Applied pressure gradient	Velocity field
	Sedimentation rate	Force	Mobility

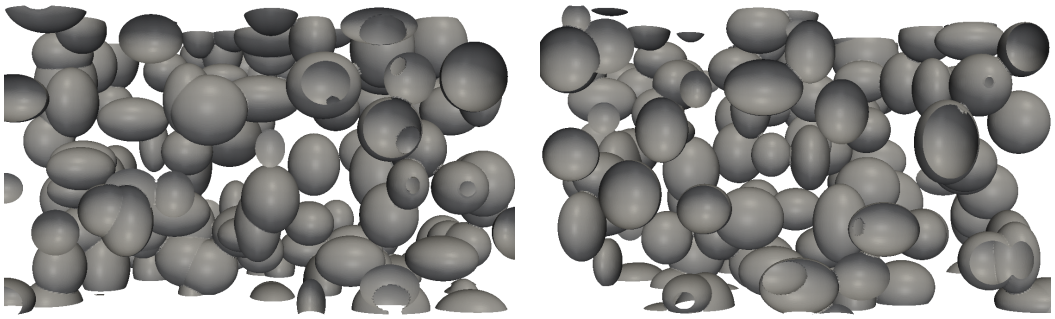
Table 3.3: Different classes of steady-state effective media problems considered here. $F \propto K_e \dot{G}$, where K_e is the general effective property, G is the average (or applied) generalized gradient or intensity field, and F is the average generalized flux field. Class A and B problems share many common features and hence may be attacked using similar techniques. Class C and D problems are similarly related to one another.

simplest level of information.

3.4.2 The geometry generation

After introducing the problem, we show how we can generate our stochastic realizations. The first step requires to build a representative microscopic model of a porous media in order to simulate fluid flow in these systems. There are many ways to obtain such model: one way could be using real sample images, for example experimentally acquired by micro-computer tomography techniques, another could be reconstructing realistically by means of suitable algorithms. The latter is the one used by Icardi *et al.* in their work [10] and it simulate the sedimentation of real three-dimensional grains, represented by convex polygonal surface meshes, generating loose sand-like structures from given particle forms and grain size distribution. A third way, the one we used in this work, could be placing point randomly in the domain until the desired porosity is reached. We just remind that the definition of porosity is the volume of grains over the total volume. This last method is less realistic and the model is far from a natural porous media but it is more interesting, from a UQ point of view, to study the effect of a random geometry. As already said, the problem is very complex and, in order to avoid the homogenization phenomenon, we to consider a right number of grains. In one hand, if we have that too many grains we will cause homogenization, therefore the multilevel algorithm would fail because the quantities, from one level to another will be *uncorrelated*. On the other hand, if we

have too few grains, we will lose the physical relevance of the problem. In Figure 3.3 we show too examples of random geometry that we use to simulate the incompressible Navier-Stokes flow at the pore scale.



(a) **(b)**

Figure 3.3: Random geometry realizations.

So, to go down the third path described above, we first have to establish the sources of randomness. In the end we consider two of them: the size together with the shape of the grains and their position in the domain.

We deal with this task in an ad-hoc algorithm written ex-novo and described entirely in Appendix A together with the whole code used for the simulations. The main function for the grain generation is `sample` and in the following we report the pseudocode.

In order to make our model more realistic, we thought and realized other improvements. First of all we add the possibility of having ellipsoids instead of simple spheres. Then we also developed a function that permits us to choose if we want an *homogeneous* or *heterogeneous porosity* that works with an acceptance-rejection algorithm.

Algorithm 4 `sample()`

$V_0 \to 2L_x \cdot 2L_y \cdot 2L_z$ (total domain volume)
$V \to 0$ (grain volume)
for $i = 1 \to$ #grains **do**
    $p \to (V_0 - V)/V_0$ (check for porosity)
    if desired porosity is reached **then**
        break
    end
    if we do not want overlapping **then**
        tries $\to 0$
        $x \sim \text{Unif}(-L_x, L_x), y \sim \text{Unif}(-L_y, L_y), z \sim \text{Unif}(-L_z, L_z)$
        μ generated from a desired distribution (Uniform, Lognormal) or fixed
        while max_tries is reached **do**
            check for overlapping
            tries $\to$ tries+1
            if tries $\equiv$ max_tries **then**
                break
            end
        end
        $V = V + \frac{4}{3}\pi\mu^3$
    else
        $x \sim \text{Unif}(-L_x, L_x), y \sim \text{Unif}(-L_y, L_y), z \sim \text{Unif}(-L_z, L_z)$
        generate μ from desired distribution (Uniform, Lognormal)
        $V = V + \frac{V}{V_0}\frac{4}{3}\pi\mu^3$ ["statistical" porosity calculation]
    end
end

3.5 Simulations

In this section we examine the performance of the multilevel Monte Carlo method in computing the expected values of some quantities of interest for different model problems in 3D. In the application with physical relevance we will focus on the expected value of the cumulative outflow from the domain on the outlet.

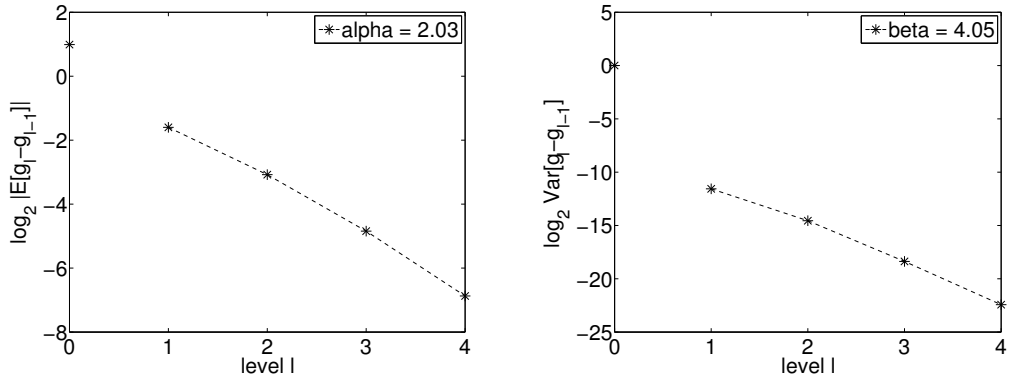
The simulations have been carried out in parallel on a Linux workstation with 22 Intel Xeon E5-2680 cores at 2.70 GHz and the CPU time for each simulation is reported case by case in the relative table of results.

3.5.1 Elliptic PDE with random forcing

The main purpose is to test our MLMC pore-scale simulation. The problem is relatively simple. It is a laplacian in a three-dimensional domain with a constant random forcing and with homogeneous Dirichlet boundary conditions.

$$\begin{cases} \Delta u(\mathbf{x}) = f(\omega), & \forall \mathbf{x} \in \Gamma \subseteq [0,1]^d \\ u(\mathbf{x}) = 0, & \forall \mathbf{x} \in \partial\Gamma \end{cases} \tag{3.5}$$

where Γ, in this case, is exactly the open cube $(0,1)^3$, $f(\omega)$ is constant random forcing uniformly distributed in $(0,100)$.



(a) α *estimate.* **(b)** β *estimate.*

Figure 3.4: Results for the QoI in the elliptic PDE with random forcing.

As quantity of interest, we consider the functional

$$g(\mathbf{u}) = \int_\Gamma \mathbf{u}(\mathbf{x})\, d\mathbf{x}$$

At each realization we call the three-dimensional finite element mesh generator (Gmsh) and the open source finite element solver GetDP.

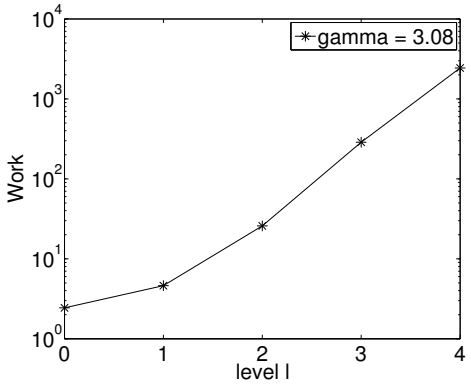


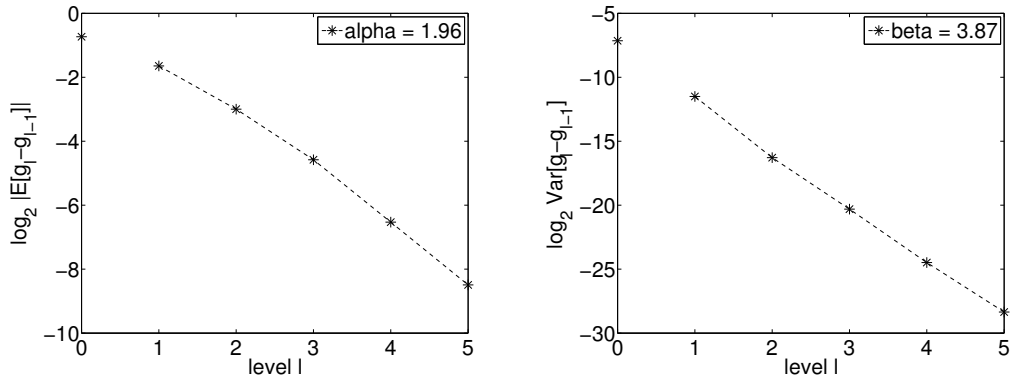
Figure 3.5: Results for the QoI in the elliptic PDE with random forcing - γ *estimate*

In order to solve the problem we are using a piecewise linear FEM so, in the same fashion of Equation 3.4, we are expecting $\alpha \approx 2$ and $\beta \approx 4$. In Figures 3.4 and 3.5 are represented the rates of α and β. For each level l we have the value of the $\mathbb{E}\left[g_l - g_{l-1}\right]$ and the corresponding error bar in the plot on the left, while in the right we have the values of $\mathrm{Var}\left[g_l - g_{l-1}\right]$ with the error bars. The number between each point represent the local rate so, in order to have an overall estimate, we should average all the value giving more importance to the last terms that are more representative of the asymptotic behavior of the problem.

As we can see, the results obtained as overall estimate are close enough to two for α and to four for β, so they are in agreement with the assumptions and they let us think that the code works and that we can rely on it for further implementations.

3.5.2 Elliptic PDE with a single random inclusion

Here we choose a problem that has not a real physical meaning, but it is our first approach to PDEs on random geometry. We can think at this example as a problem where the domain is divided in two subdomains randomly and one of those is a sphere that can at most touch the original boundaries.



(a) α *estimate.* **(b)** β *estimate.*

Figure 3.6: Results for the elliptic PDE with a single random inclusion.

The problem is again a three-dimensional laplacian with constant forcing and homogeneous Dirichlet boundary conditions

$$\begin{cases} \Delta u(\mathbf{x}) = f, & \forall \mathbf{x} \in \Gamma(\omega) \subset [0,1]^d \\ u(\mathbf{x}) = 0, & \forall \mathbf{x} \in \partial\Gamma(\omega) \end{cases} \tag{3.6}$$

where $\Gamma(\omega)$ is the open subset of $[0,1]^3$ with a sphere shaped hole randomly placed in the sense that, given a fixed radius r, the position (X,Y,Z) is represented by three RV uniformly distributed as $X \sim \mathcal{U}(-(0-r), 1-r)$, $Y \sim \mathcal{U}(-(0-r), 1-r)$ and $Z \sim \mathcal{U}(-(0-r), 1-r)$.

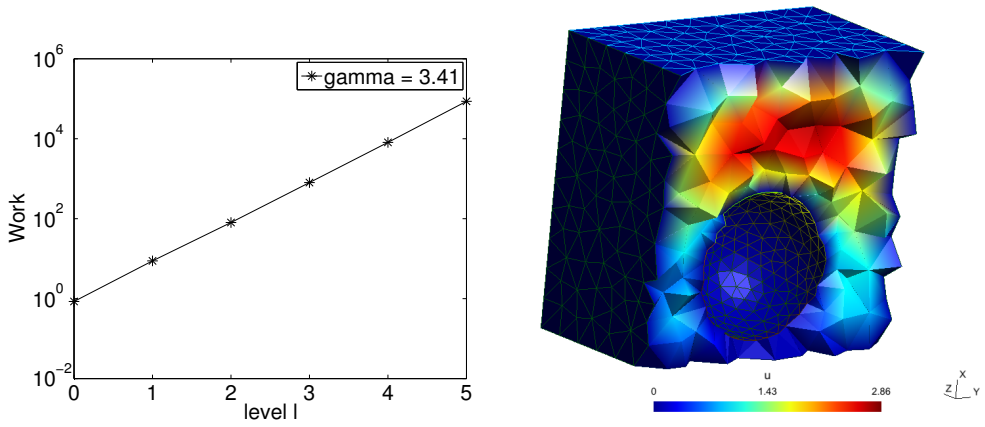
Results of the simulation, *Estimator* = 1.10			
MLMC stat. error	$7.0 \cdot 10^{-3}$	MLMC stat. error %	0.0064
StdMC stat. error	$2.3 \cdot 10^{-1}$	Stat. err. gain factor	33
MLMC Work	$3.9 \cdot 10^{5}$ ($\sim$ 5 d)	StdMC Work	$2.0 \cdot 10^{7}$ ($\sim$ 234 d)

Table 3.4: Results for the elliptic PDE with random forcing.

As quantity of interest, similarly to the previous case, we consider the functional

$$g(\mathbf{u}) = \int_{\Gamma(\omega)} \mathbf{u}(\mathbf{x})\, d\mathbf{x}$$

As in our first instance with the new code, the blackbox solver is GetDP and the mesh is built with Gmsh and similarly to the previous case, we have $\alpha \approx 2$ and $\beta \approx 4$. Moreover, this time in the domain we have two different conductivities corresponding to the different subdomains. In Figure 3.7(b) we report a realization of the random inclusion and, for sake of clearness, we show a section of the domain in correspondence of the inclusion itself while, in Figure 3.7(a), the estimate of $\gamma \approx 3.5$ is shown and it is a bit higher than the expected 3.



(a) γ *estimate.* **(b)** *Section of the domain.*

Figure 3.7: Results for the elliptic PDE with a single random inclusion.

In Table 3.4 we report the main results of the simulation. We will now describe the outputs that will be reported in same way for all the following cases. The first two quantities are the statistical error discussed in Chapter 2 and the relative percentage with respect to the estimator. In the middle row we make a first comparison with the standard Monte Carlo method while keeping the bias error fixed. We recall that we have previously defined the MSE as $|\mathbb{E}\left[g\right] - Y|$, but, as we did in Chapter 2, it can controlled separately by splitting it in bias and statistical error, defined as

$$\text{bias } = |\mathbb{E}\left[g - Y\right]|, \qquad \text{stat. err. } = |\mathbb{E}\left[Y\right] - Y|$$

respectively.

The statistical error of a single level is computed as

$$\text{StdMC stat. err } = c_0 \sqrt{\frac{\text{Var}\left[g_L\right]}{M_L}} \tag{3.7}$$

and it represents, the statistical error that would be achieved with only the last single level. In order to give an idea of the improvement we also report the gain factor that is simply the ratio between the statistical of the single level and the original statistical error. Finally the last two quantities are the MLMC work and the standard Monte Carlo one. The previous is computed directly as the sum of the CPU time in each level

$$\sum_{l=0}^{L} M_l \cdot W_l$$

while the latter is an estimate computed as

$$M_{MC} \cdot W_L$$

where the number of Monte Carlo samples is calculated using Equation 3.7, replacing the single level statistical error with the standard one and then solving for M_L. In other words, these last two quantities represent a comparison of the work for the same level of error.

As we can see from the Table 3.4, even for this simple the standard Monte Carlo method is clearly outperformed. In Table 3.7, instead, we make a statement on the theoretical complexity.

3.5.3 Diffusion on a randomly perforated domain

In this third case, we consider a problem of more physical relevance. Even though we cannot assume to deal with a porous medium, in this application we apply a pressure gradient and we compute the diffusive flux in the outlet. The formulation becomes

$$\Delta u(\mathbf{x}) = f, \qquad \forall \mathbf{x} \in \Gamma(\omega) \subset [0,1]^d \tag{3.8}$$

and the boundary is partitioned

$$\begin{cases} \text{no-slip:} \quad \mathbf{u} = 0, \quad \nabla_n p = 0 & \text{on} \quad I_g \\ \text{inlet:} \quad \nabla_n \mathbf{u} = 0, \quad p = p_i(\mathbf{x}) & \text{on} \quad I_i = \{\mathbf{x} : x_1 = 0\} \\ \text{outlet:} \quad \nabla_n \mathbf{u} = 0, \quad p = p_o(\mathbf{x}) = 0 & \text{on} \quad I_o = \{\mathbf{x} : x_1 = 1\} \\ \text{symmetry:} \quad \nabla_n u_t = 0, \quad u_n = 0, \quad \nabla_n p = 0 & \text{on} \quad I_s = \{\mathbf{x} : x_2, x_3 = 0\} \end{cases}$$

Our quantity of interest is the effective diffusive flux of the sample given by

$$g(\mathbf{u}) = \frac{\int_0^1 \mathbf{u}(\{1,0\}, x_2, x_3)\, dx_2\, dx_3}{\Delta p}.$$

where $\Delta p = \langle p(0, x_2, x_3)\rangle - \langle p(1, x_2, x_3\rangle$ is the mean pressure gradient between inlet and outlet.

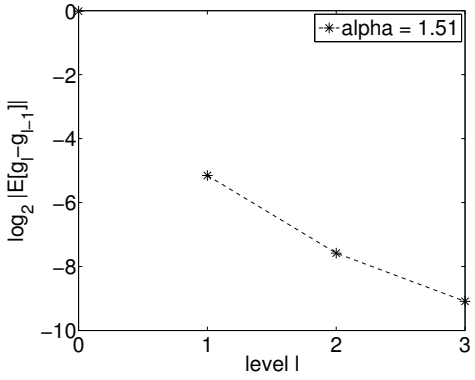
The domain is similar to the previous one, it is divided into two subdomains with different conductivities, one and zero. The inclusions are several this time and they are randomly placed in each and every single realization with Algorithm 4 but, in this case, there is no "porosity" to be reached. In fact, the number of inclusions is fixed and small.

Results of the simulation, *Estimator* = 0.98			
MLMC stat. error	$1.3 \cdot 10^{-4}$	MLMC stat. error %	0.0001
StdMC stat. error	$4.6 \cdot 10^{-3}$	Stat. err. gain factor	35
MLMC Work	$6.6 \cdot 10^{4}$ ($\sim$ 18 h)	StdMC Work	$7.0 \cdot 10^{5}$ ($\sim$ 8 d)

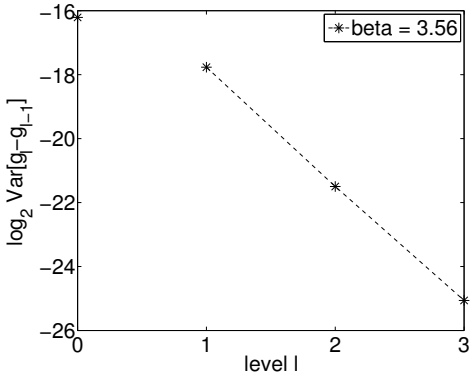
Table 3.5: Results for the diffusion in a random domain.

At each realization we call the three-dimensional finite element mesh generator (Gmsh) and the open source finite element solver GetDP.

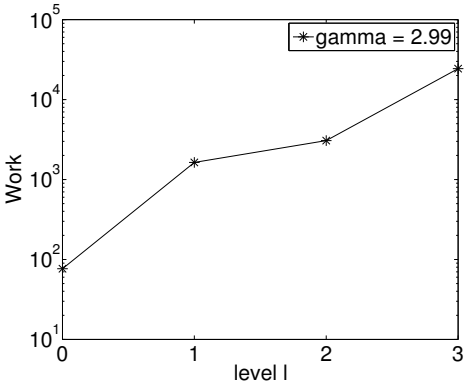
In figure 3.8(d) we can see a section of the cubic domain where the solution, the stationary field, is represented. In the other 3 plot of Figure 3.8 we estimate α, β and γ. The results of the statistics are reported in Table 3.5 while in Table 3.7we report α, β and γ and we make a statement on the theoretical complexity.



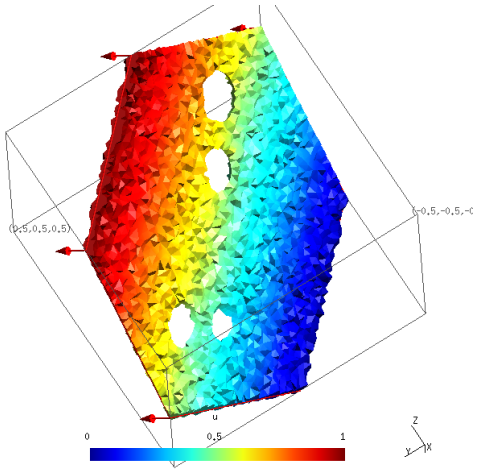
(a) *α estimate.*



(b) *β estimate.*



(c) *γ estimate.*



(d) *Section of the domain.*

Figure 3.8: Results for the diffusion on a randomly perforated domain.

3.5.4 Pore-scale Navier-Stokes

In this paragraph we consider the adimensional steady incompressible Navier-Stokes equation[3]

$$\begin{cases} \mathbf{u}(\mathbf{x};\omega)\cdot\nabla\mathbf{u}(\mathbf{x};\omega)+\nabla p(\mathbf{x};\omega)-\frac{1}{Re}\nabla\cdot(\nabla\mathbf{u}(\mathbf{x};\omega))=0, \\ \nabla\cdot\mathbf{u}=0 \end{cases} \qquad \forall \mathbf{x}\in\Gamma(\omega)\subset[0,1]^d,$$

with the Reynolds number $Re=\frac{UL}{\nu}$, and the pressure p, and velocity $\mathbf{u}$ being the stochastic solutions from which we seek to approximate some QoI. The domain $\Gamma(\omega)\subset[0,1]^3$ is a subset of the unit cube with several inclusions that, in this case, can represent the pores of a porous medium. The algorithm for the geometry generation is exactly the one in Algorithm 4. The radius μ of the grains is sampled from a uniform distribution as well as the position (X,Y,Z) in the space until a fixed desired porosity is reached.

Partitioning the boundary $\partial\Gamma(\omega)$ in subregions I_i, the following boundary conditions are imposed:

$$\begin{cases} \text{no-slip:} \quad \mathbf{u}=0, \quad \nabla_n p=0 & \text{on} \quad I_g \\ \text{inlet:} \quad \nabla_n\mathbf{u}=0, \quad p=p_i(\mathbf{x}) & \text{on} \quad I_i=\{\mathbf{x}: x_1=0\} \\ \text{outlet:} \quad \nabla_n\mathbf{u}=0, \quad p=p_o(\mathbf{x})=0 & \text{on} \quad I_o=\{\mathbf{x}: x_1=1\} \\ \text{symmetry:} \quad \nabla_n u_t=0, \quad u_n=0, \quad \nabla_n p=0 & \text{on} \quad I_s=\{\mathbf{x}: x_2,x_3=0\} \end{cases}$$

where ∇_n is the derivative in the normal direction and u_t and u_n are respectively the tangential and normal direction of the velocity with respect to the boundary.

Our quantity of interest is the effective horizontal permeability of the sample given by

$$g(\mathbf{u})=\frac{\int_0^1 \mathbf{u}(\{1,0\},x_2,x_3)\,dx_2\,dx_3}{\Delta p}.$$

where $\Delta p=\langle p(0,x_2,x_3)\rangle-\langle p(1,x_2,x_3\rangle$ is the mean pressure gradient between inlet and outlet.

Numerical solution realizations

The three dimensional incompressible steady Navier-Stokes eqution were solved at each step of the MLMC routines with black-box solver `simpleFoam` also coming from the open-source code `OpenFOAM`. The equations are discretize with the Finite Volume Method. A second-order central difference schemes with limiters to avoid oscillations was used for spatial discretization

[3] The Stokes flow equations are obtained by simply removing the first non-linear advection term in the first equation

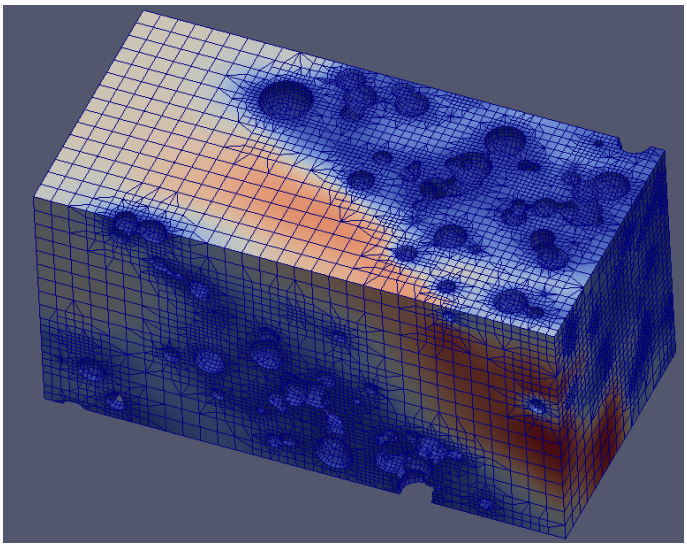


Figure 3.9: Mesh of a geometry realization.

and the SIMPLEC scheme was used to overcome the pressure-velocity coupling problem. The whole domain was studied with a fixed hydraulic head drop between inlet and outlet and with symmetric conditions on the lateral boundaries. This means that the resulting main flow is directed among the x-axis and there is no flow escaping from lateral boundaries.

Typically, we are not capable of deriving solution realizations $u_\ell(\omega)$ by analytical means. Numerical approximations of the solution realizations can be constructed by first approximating the geometry $\Gamma_\ell(\omega)$ by a numerical method representation $\overline{\Gamma}_\ell(\omega)$, thereafter using a black box numerical solver, in our case `OpenFOAM`, to generate a numerical solution realization $\overline{u}_\ell(\omega)$ from the input geometry $\overline{\Gamma}_\ell(\omega)$, and at the end compute the QoI $g(\overline{u}_\ell(\omega))$. For implementational purposes we will assume there exists an $\alpha > 0$ such that

$$\left|\mathbb{E}\left[g(u) - g(\overline{u}_\ell)\right]\right| = \mathcal{O}\left(2^{-\alpha\ell}\right), \tag{3.9}$$

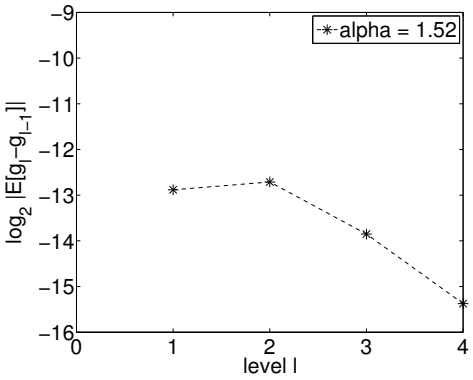
and a $\gamma > 0$ such that the cost of computing/generating a numerical solution realization pair $(\overline{u}_{\ell-1}(\omega), \overline{u}_\ell(\omega))$ and computing its QoI has the following asymptotic bound

$$C_\ell := \mathbb{E}\left[\text{Computational cost}(\Delta_\ell g(\cdot)\right] = \mathcal{O}\left(2^{\gamma\ell}\right),$$

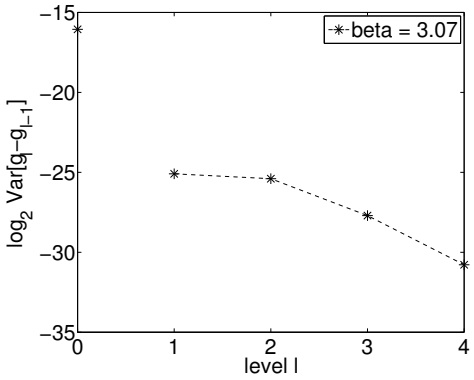
where we denote

$$\Delta_\ell g(\omega) = g(\overline{u}_\ell(\omega)) - g(\overline{u}_{\ell-1}(\omega)).$$

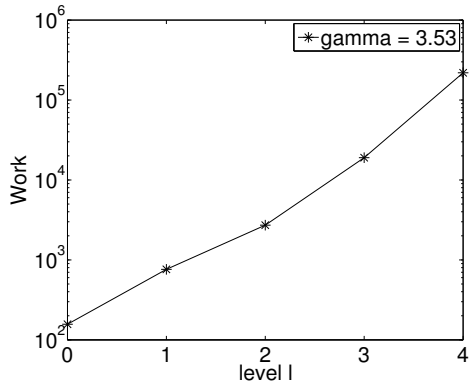
In Figure 3.10 we estimate the weak and the strong error rates. In this case, as stated above, we are using second-order schemes so we are expecting an $\alpha \approx 2$. However, since our grid is not



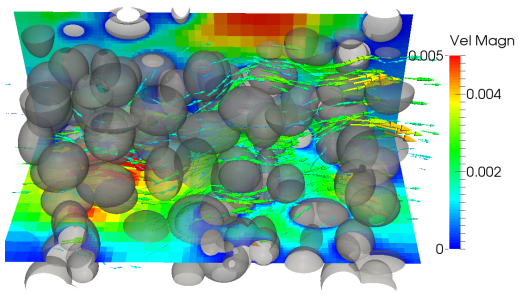
(a) α *estimate.*



(b) β *estimate.*



(c) γ *estimate.*



(d) *Deterministic solution with streamlines.*

Figure 3.10: Results for the incompressible Navier-Stokes flow simulation..

structured, the 1.5 obtained as rate is not a bad result. The strong error rate is also not bad and it is 3. The rate of the work is good considering that we are dealing with a three-dimensional problem and the grid is highly non-uniform, as we could appreciate in Figure 3.9 where we can see that there are refinements in correspondence of the grains. In particular, in Figure 3.9, we also want to show the an example of the heterogenous porosity that we were talking about in section 3.4.2. The building of our grid was made by the mesh utility `snappyHexMesh`, native to the open-source package `OpenFOAM`. This operation was performed in two steps. First, a structured, cartesian mesh was created in the fluid portion of the domain, in order to minimize average non-orthogonality and skewness. The second step consisted in modifying the mesh by means of relocating boundary vertices, resulting in a body fitted mesh.

Results of the simulation, *Estimator* = $2.1753 \cdot 10^{-3}$			
MLMC stat. error	$1.3 \cdot 10^{-4}$	MLMC stat. err. %	0.0617
StdMC stat. error	$4.9 \cdot 10^{-2}$	Stat. err. gain factor	362
MLMC Work	$4.6 \cdot 10^{5}$ (~ 5 d)	StdMC Work	$1.7 \cdot 10^{7}$ (~ 191 d)

Table 3.6: Results for the incompressible Navier-Stokes flow simulation.

In Table 3.6 we report the results of the simulation and, similarly to the other cases, the advantages of the MLMC method is clear. In this case we can see how the standard Monte Carlo is not only outperformed, but also it would be impossible to realize in a reasonable time. In Table 3.7we report α, β and γ and we make a statement on the theoretical complexity.

3.6 Conclusions

The work done in this thesis attempts to emphasize the conceptual simplicity of the multilevel approach; in essence it is simply a recursive control variate strategy, using cheap approximations to some random output quantity as a control variate for more accurate but more costly approximations. In practice, the challenge is to develop a tight coupling between successive approximation levels, to minimize the variance of the difference in the output obtained from each level. After an analysis of the results of the previous applications to fluid dynamics, we can state two important conclusions regarding the Multilevel Monte Carlo method.

First, we used the Multilevel Monte Carlo method for very complex problems with different black-box solvers like `OpenFOAM` and GetDP and we proved it to be flexible and modular, even though is not trivial, because the computational costs of some applications are still considerable. The algorithm can be used with application-specific routines and, in most of the cases, it works and it is robust.

With the numerically observed values for α, β and γ it is also possible to compare the theoretically predicted costs given by Theorem 4 for each case we studied and we do this in Table 3.7. We see from Table 3.7 that asymptotically the MLMC leads to a huge improvement over the standard MC for all the cases.

Case	α	β	γ	MLMC compl.	StdMC compl.
Rand. forcing	2	4	3	$O(\varepsilon^{-2})$	$O(\varepsilon^{-7/2})$
Single rand. incl.	2	4	3.5	$O(\varepsilon^{-2})$	$O(\varepsilon^{-15/4})$
Rand. diffusion	1.5	3.5	3	$O(\varepsilon^{-2})$	$O(\varepsilon^{-4})$
Pore-scale N-S	1.5	3	3.5	$O(\varepsilon^{-7/3})$	$O(\varepsilon^{-13/3})$

Table 3.7: Predicted cost of the MLMC estimator given by Theorem 4 to achieve a MSE of ε^2 compared to the cost of the standard MC estimator given by Equation 2.4.

To conclude, in this thesis we successfully applied the Multilevel Monte Carlo algorithm to elliptic PDEs with random parameters. The results clearly show the advantage of using the MLMC estimator over a standard MC estimator for this type of model and for the quantities of interest considered. They further show that the gain of the MLMC estimator is not limited to smooth or easy problems but also for very complex ones. The improvements are in fact considerable when the solver cost is relevant (for 3D problem $\gamma \geq 3$), or in cases where the discretization error is large.

APPENDIX A

Multilevel Monte Carlo Pore-Scale Code

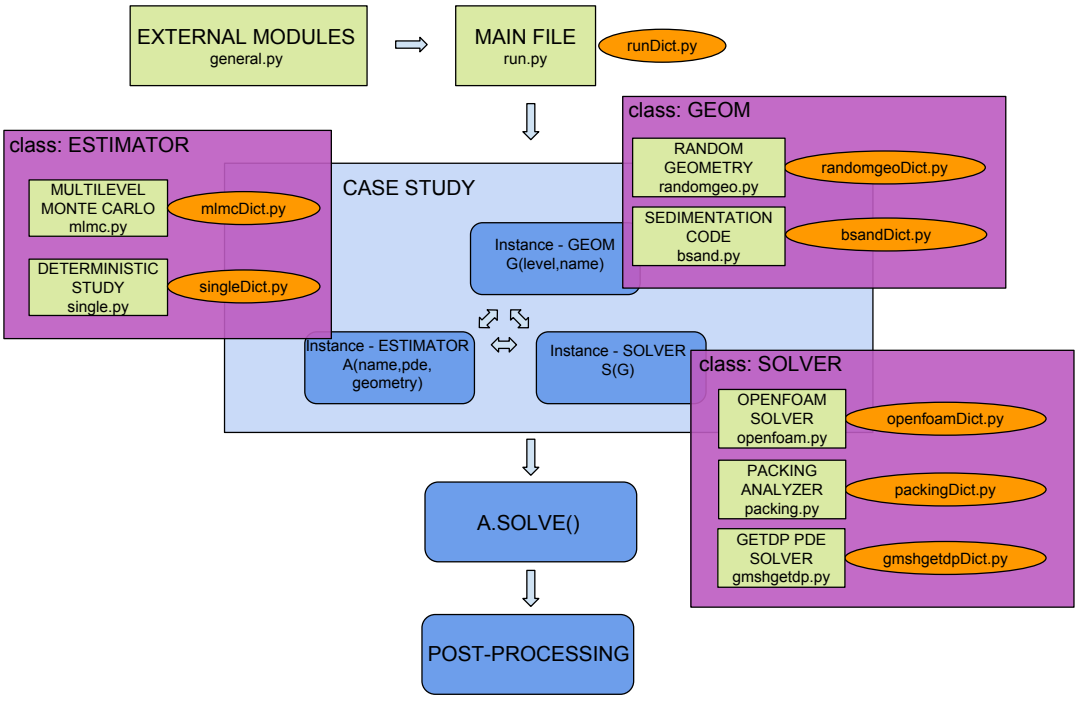


Figure A.1: Pore-Scale Code.

In this Appendix we want to give an idea on the main purposes and structure of the code

we used for the MLMC simulations presented in this thesis. In Figure A.1 we show a schematic representation of its main features that we will explain in details below. The code [27] has been structured as a wrapper, developed in Python, containing calls to external sub-modules. An object-oriented programming paradigm has been used, by creating abstract classes of different types whose multiple instances are then created.

A.1 Main purpose

After understanding the potentialities of multilevel Monte Carlo and after a deep study of Giles' algorithm we extended the code in order to apply the method to more complex problems such as the ones described in Chapter 3. To do this, we need to combine existing solvers of different types and generate complex pre- and post- processing routines. The code has been structured as a "general-purpose" MLMC estimator for complex computational problems therefore with a high level of abstraction and flexibility.

The basis of the code was initially thought to solve a finite number of deterministic problems like the ones in [26], where the fluid flow and the solute transport through porous media are described. Since we had in mind something that could work in a wider spectrum of problems and with different approaches, we extended it also for statistical estimation of random PDE, keeping the physical complexity of the problems.We ended up with a code that can deal with deterministic and statistical problems that can solve different PDE model equations. This is achieved by calling external solvers such as `OpenFOAM` , GetDP and Fenics in order to solve problems like Steady State Navier-Stokes, two-phases compressible flows, two-phases incompressible flows, Darcy, Laplacian, etc.

A.2 Structure

The main script of the whole program is `run.py` where all the external modules are imported, the main settings of the estimation are loaded and the main object, containing (called estimator) is created and solved. All the settings are imported from dictionary files, such as `runDict.py`. Every part of the program, in fact, is associated with a dictionary where the parameters are stored and can be easily configured by the user, as we can clearly see in Figure A.1. As we can also see from Figure A.1, there are three classes that characterize the three main elements of each problem. Each of these elements can be loaded by different files, according to the problem we want to solve

From the first one, the **Estimator** class, we can choose the approach of problem. In fact, if we want to estimate a quantity of interest with the multilevel Monte Carlo method we create an instance with `mlmc.py`, on the other hand, if we want to find a deterministic solution or a fixed parametric study of a given problem we create an instance with `single.py`. In the former the Giles' multilevel algorithm and other similar algorithms are implemented. They can be adaptive, like the original one, or we can set *a priori* some parameters like the initial number of sample and number of levels. These parameters, together with the confidence (c_0), the tolerance (ε or TOL) and the splitting between the bias and the statistical error, can be set in `mlmcDict.py` WIth the latter instead, a single deterministic simulations can be run.

After the estimator has been created and set up, it is solved by running the selected algorithm, implemented in the class itself. To do this usually a second class is used, called **Geom** class that store all the information of the geometry. For each realization an instance of the Geom class is created. Two kind of geometries can be created. With `randomgeo.py` the geometry is constructed, as we have already well seen in Chapter 3, as a realization of two random variables, one for the position of the grains (if we deal with a porous medium) or the position of the inclusion (if we do not deal with a physically relevant problem), and one for the characteristic dimension of the grain/inclusion, i.e. the radius. With `bsand.py` instead, a realistic porous medium geometry can be created. In this latter case, is simulated the sedimentation of the real three-dimensional grains, represented by convex polygonal surface meshes, generating loose sand-like structures from given particle forms and grain size distributions. The actual grain shape and grain size distributions are obtained respectively by two-dimensional scanning electron microscopy (SEM) scans and static-light scattering measurements carried out on standard sand samples.

A third and last is **Solver** class. Has to do with the all the settings and routines specific for each solver. A geometry is passed to construct a solver object on top of it. The solver is defined in three files, `openfoam.py`, `gmshgetdp.py` and `packing.py`, in order to define which code to use to solve our problem. The first has the obvious connection with the open-source code `OpenFOAM`[1]. In this case, each time a sample is realized and the geometry is built, it creates the mesh with the utility `snappyHexMesh` and it calls the right solver suitable for the problem we are trying to solve (for our problems `simpleFoam`, the steady-state solver for incompressible and turbulent flows). The second solver, instead, is linked to Gmsh[2], a three-dimensional

[1] website - http://www.openfoam.com

[2] website - http://geuz.org/gmsh/

finite element mesh generator, and GetDP[3] ("General environment for the treatment of Discrete Problems"), an open-source scientific software environment for the numerical solution of integro-differential equations, open to the coupling of physical problems (electromagnetic, thermal, etc.) as well as of numerical methods (finite element method, integral methods, etc.) which can deal with such problems of various dimensions (1D, 2D or 3D) and time states (static, transient or harmonic). The last but not least, `packing.py` is not aPDE solver, but it is though to run statistical analysis' on the geometry realizations once they are generated.

After each geometry and solver instances are created, the problem is solved, the quantities of interest are stored in the estimator object and adaptively used for estimation. When the algorithm stops results are plotted and a post-processing analysis can be conducted. It is important to highlight that, being the Monte Carlo samples independent, the code can be parallelized at two levels: the level of the MLMC estimator and the level of the single solver. Both of these approaches have been used and combined.

A.3 Final statements

During the whole process of the thesis development we run many test cases to test and debug the code and, in the end, we can say to have proved its effectiveness in important applications.

However, due to the complexity of the problems at hand, there is still space to many further improvements related to performance, parallelization, analysis of convergence and non-asymptotic behaviors. Also the extension to different problems and the linking to other external software would be something extremely interesting for future works.

[3] website - http://www.geuz.org/getdp/

Bibliography

[1] S. Asmussen, P. W. Glynn *Stochastic Simulation, Algorithms and Analysis*, Springer, 2007.

[2] I. Babuška, F. Nobile, R. Tempone *A stochastic collocation method for elliptic partial differential equations with random input data*, SIAM Review 52(2), 2010.

[3] I. Babuška, R. Tempone, G. Zouraris *Galerkin finite element approximations of stochastic elliptic partial differential equations*, SIAM Journal on Numerical Analysis 42(2), 800-825, 2004.

[4] J. Baldeaux, M. Gnewuch, *'Optimal randomized multilevel algorithms for infinite-dimensional integration on function spaces with ANOVA-type decomposition*, SIAM Journal of Numerical Analysis 52(3), 1128–1155, 2014.

[5] A. Barth, C. Schwab, N. Zollinger *Multilevel Monte Carlo finite element method for elliptic PDEs with stochastic coefficients* Numer. Math. Online First, 2011.

[6] J. Bear *Dynamics of fluids in porous media*, Dover Publications, 1988.

[7] C. Canuto *Notes on Partial Differential Equation*, 2009.

[8] C. Canuto, S. Berrone *Notes on Numerical Methods for Partial Differential Equation*, 2011.

[9] K. A. Cliffe, M. B. Giles, R. Scheichl, A. L. Teckentrup *Multilevel Monte Carlo methods and application to elliptic PDEs with random coefficients*, Springer-Verlag 2011.

[10] N. Collier, A. Haji-Ali, F. Nobile, E. von Schwerin, R. Tempone, *A Continuation Multilevel Monte Carlo algorithm*, 2014.

[11] J. Dick, M. Gnewuch *Infinite-dimensional integration in weighted Hilbert spaces: anchored decompositions, optimal deterministic algorithms, and higher order convergence*, Foundation of Computational Mathematics 184, 111–145, 2014a

[12] J. Dick, M. Gnewuch *Optimal randomized changing dimension algorithms for infinite-dimensional integration on function spaces with ANOVA-type decomposition*, Journal of Approximation Theory 184, 111–145, 2014b

[13] J. Dick, F. Y. Kuo, I. H. Sloan *High-dimensional integration: The quasi-Monte Carlo way*, Acta Numerica / Volume 22 / May 2013, pp 133-288.

[14] J. H. Ferziger, M. Perić *Computational Methods for Fluid Dynamics*, Springer-Verlag, 2002

[15] M. B. Giles, *Multilevel Monte Carlo path simulation*, Oxford, UK.

[16] M. B. Giles, *Multilevel Monte Carlo methods*, Oxford (2015), UK.

[17] M. B. Giles, B. Waterhouse *Multilevel quasi-Monte Carlo path simulation*, Advanced Financial Modelling, Radon Series on Computational and Applied Mathematics, 165-181, 2009.

[18] P. Glasserman *Monte Carlo Methods in FInancial Engineering*, Springer, 2003.

[19] M. Gunzburger, C. Webster, G. Zhang *Stochastic finite element methods for partial differential equations with random input data*, Acta Numerica 23, 2014.

[20] A. Haji-Ali, F. Nobile, E. von Schwerin, R. Tempone, *Optimization of mesh hierarchies in Multilevel Monte Carlo samplers*, 2014.

[21] S. Heinrich, *Monte Carlo complexity of global solution of integral equations*, Journal of Complexity 14(2), 151-175, 1998.

[22] S. Heinrich, E. Sindambiwe *Monte Carlo complexity of parametric integration*, Journal of Complexity 15(3), 317-341, 1999.

[23] A. Kebaier *Statistical Romberg extrapolation: a new variance reduction method and applications to options pricing*, Annals of Applied Probability 14(4), 2681-2705, 2005.

[24] F. Y. Kuo, C. Schwab, I. H. Sloan *'Multi-level quasi-Monte Carlo finite element methods for a class of elliptic partial differential equations with random coefficients*, ArXiv preprint:, 2012.

[25] G. Iaccarino, T. Magin, *Short Course on Uncertainty Quantification*, Stanford CA, USA.

[26] M. Icardi, G. Boccardo, T. Tosco, D. L. Marchisio, R. Sethi, *Pore-scale simulation of fluid flow and solute dispersion in three-dimensional porous media*, 2014.

[27] M. Icardi, R. Tempone, *Multilevel Monte Carlo Applications to Geometrical and Differential Problems on Random Geometry*, (in preparation).

[28] O. P. Le Maître, O. M. Knio, *Spectral Methods for Uncertainty Quantification*, Springer

[29] C. Lemieux, *Monte Carlo and Quasi-Monte Carlo Sampling*, Springer, 2009.

[30] G. Migliorati, F. Nobile, E. von Schwerin, R. Tempone, *Analysis of the discrete L^2 projection on polynomial spaces with random evaluation*, 2011.

[31] B. Niu, F. Hickernell, T. Müller-Gronbach, K. Ritter *Deterministic multi-level algorithms for infinite-dimensional integration on* $\mathbb{R}^N$, Journal of Complexity 27(3-4), 331-351, 2010.

[32] F. Nobile, R. Tempone, *Mathematical and Algorithmic Aspects of Uncertainty Quantication* Summer School at University of Texas, slides, 2014.

[33] OpenCFD *The Open Source CFD Toolbox, User Guide*, OpenCFD (ESI), 2013.

[34] M. Putko, A. Taylor, P. Newmann, L. Green *Approach for input uncertainty propagation and robust design in CFD using sensitivity derivatives*, Journal of Fluids Engineering 124(1), 60-69, 2002.

[35] D. S. Sivia, *Data Analysis, a Bayesian Tutorial*, Oxford University Press, 2006.

[36] A. Stuart, *Uncertainty Quantification in Bayesian Inversion*, 2010.

[37] S. Torquato, *Random Heterogeneous Materials: Microstructure and Macroscopic Properties*, Springer-Verlag, 2002.

[38] D. Xiu, G. Karniadakis *The Wiener–Askey polynomial chaos for stochastic differential equations*, SIAM Journal on Scientific Computing, 2002.