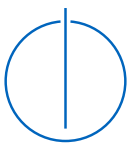
DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Informatics

Multifidelity Monte Carlo Sampling in Plasma Microturbulence Analysis

Julia Konrad



DEPARTMENT OF INFORMATICS

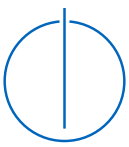
TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Informatics

Multifidelity Monte Carlo Sampling in Plasma Microturbulence Analysis

Multifidelity Monte Carlo Sampling in der Plasmamikroturbulenzanalyse

Author:	Julia Konrad
Supervisor:	Prof. Dr. Hans-Joachim Bungartz
Advisors:	Ionuț-Gabriel Farcaș, Dr. Tobias Neckel
Submission Date:	August 16, 2019



I confirm that this bachelor's thesis is my own work and I have documented all sources and material used.

Garching, August 16, 2019 Julia Konrad

Acknowledgments

First of all, I would like to express my gratitude to my advisor Ionuț-Gabriel Farcaș for his patience, always having an open ear for any questions I had, and continuously providing invaluable advice and feedback during my work on this project. Moreover, I would like to thank Tobias Neckel for his guidance in the inital phases of this thesis.

I also want to thank Prof. Hans-Joachim Bungartz for giving me the opportunity to work on this thesis at the Chair of Scientific Computing.

Finally, a big thank you to my family, my parents, my brothers. I could not be doing what I am doing without any of them.

Abstract

In this work, we apply multifidelity Monte Carlo sampling, a technique for uncertainty quantification, to two test cases from plasma microturbulence analysis, which is an important field of research into fusion power. Multifidelity Monte Carlo sampling aims to increase the accuracy of standard Monte Carlo estimators for statistical moments of model output based on uncertain input parameters. Standard Monte Carlo sampling estimates statistics by evaluating the underlying model for a number of samples and computing estimators from the results. In order to obtain acceptable precision, a prohibitive amount of samples is often required and simulations become computationally infeasible, especially if the underlying model is expensive to evaluate. To overcome this shortcoming, multifidelity Monte Carlo sampling distributes the model evaluations between the model under consideration and one or more computationally less expensive surrogate models. The obtained model evaluations are then combined into estimators using a control variate approach.

In our test scenarios, we consider the Cyclone Base Case, a popular benchmark in the analysis of plasma turbulence. Multifidelity Monte Carlo sampling is applied to a three-dimensional and an eight-dimensional version of this scenario. For our simulations, we use the plasma microturbulence code GENE and construct a sparse grid approximation and a machine learning surrogate. In both test cases, we find that the application of multifidelity Monte Carlo sampling leads to a reduction of the estimators' mean squared error by orders of magnitude. Moreover, we show that combining different low-fidelity models can further decrease the error.

Contents

1. Introduction

In this section, general concepts about the methodologies examined in this work and the physical application under consideration are introduced. Section 1.1 summarizes the fields of uncertainty quantification, multifidelity methods and plasma microturbulence analysis, and explains why we study these areas here. In Section 1.2 we describe the specific work done in this thesis. Section 1.3 reviews previous efforts in the analysis of plasma microturbulence using UQ approaches and in the research of multifidelity methods. We outline the following chapters in Section 1.4.

1.1. Motivation

1.1.1. Uncertainty Quantification

One area within the field of scientific computing that has recently been gaining increasing attention is that of *uncertainty quantification* (UQ). This discipline is based on methodologies from probability theory, statistics and applied mathematics. A broad definition of UQ is given in [32] as "the science of identifying, quantifying, and reducing uncertainties associated with models, numerical algorithms, experiments, and predicted outcomes or quantities of interest".

Whenever computer models are used to describe the behaviour of physical systems, there is likely some uncertainty involved. Uncertainty can for instance be introduced by errors made in the measurement of real-world physical parameters that act as inputs for the model. Due to e.g. instrument precision, the accuracy of inputs can often only be guaranteed within a certain error margin. Another source of uncertainty are the numerical methods used to model the system. In order to be able to represent input and output domains, they often need to be discretized. Because of this, any model outputs are really only approximations of the true quantities. Furthermore, the true physics of a process can in some cases be difficult to model, in other situations might lead to computationally expensive models or is simply not fully understood. Therefore, the structure of a given model often represents a simplified version of reality and inherently leads to uncertainties with regard to the real-world system under consideration [32]. Uncertainties introduced by any of these sources into the modelling of physical phenomena need to be studied. In order to fully understand results obtained

from simulations processes that are subject to any form of uncertainty, it is important to assess the effect of uncertainty on the model and its output.

In the current work, we focus on uncertainties introduced by input parameters. Quantifying the effect they have on the output of a model is important in order to understand the information given by the model better. The area of UQ dealing with this type of problem is called *uncertainty propagation*. One task in this discipline might be identifying the expected value of the model output for a given set of input parameters and quantifying the confidence of this prediction by determining the variance of the output.

Monte Carlo sampling is a standard technique in uncertainty propagation for estimating the mean and variance of stochastic model outputs based on uncertain input. In the first step, (pseudo-)random samples of the input parameters are drawn based on an assumption on their probability density function. The given model is then evaluated at each sample and the corresponding ensemble of model evaluations is used to estimate statistics such as mean and variance. The convergence of this algorithm for mean estimation using n i.i.d samples lies in $\mathcal{O}(\frac{1}{\sqrt{n}})$ [17]. On the one hand, this rate does not depend on the number of input dimensions which makes Monte Carlo sampling attractive for dealing with high-dimensional problems. On the other hand, the convergence is slow as it decreases only with $\frac{1}{\sqrt{n}}$. Intuitively, this means that to gain one more digit of precision, about ten times more effort needs to be invested. Therefore, in many cases, evaluating the model for the number of samples that are needed to achieve results of an acceptable accuracy can be too computationally expensive.

1.1.2. Multifidelity Monte Carlo Sampling

To decrease the cost of Monte Carlo sampling, several other methods can be employed, e.g. control variates or importance sampling [13]. The objective of these techniques is to reduce the variance of Monte Carlo estimators. An estimator's accuracy is directly influenced by its variance. Therefore, decreasing this variance leads to an increased precision in an estimator that uses a given number of samples. In this work, we focus on *control variates* as a variance reduction technique which we employ in the context of *multifidelity Monte Carlo sampling* (MFMC).

In the simulation of physical phenomena, there are often several models available that describe the given scenario, typically varying in the precision of their output and their computational cost. Multifidelity methods make use of such models and exploit their differences. In particular, multifidelity Monte Carlo sampling tries to speed up the convergence of the standard Monte Carlo sampling process by evaluating not only the accurate and expensive *high-fidelity model* but also employing one or more *low-fidelity models* of a lower accuracy which are usually computationally cheaper. By

distributing some of the samples to less expensive models, more model evaluations can be obtained in a given time frame. The accuracy of the final result can be guaranteed by the evaluations stemming from the original model.

1.1.3. Plasma Microturbulence Analysis

The real-world application under consideration here is that of *plasma microturbulence analysis*. Since the first experiments were conducted in the 1930s, power generation by nuclear fusion has been steadily researched until the present [14]. Current nuclear reactors all harvest power generated by nuclear fission, the splitting of an atom's nucleus. Fusion, on the other hand, is the process of two nuclei combining into a larger atom. This releases large amounts of energy. In contrast to fission, nuclear fusion does not generate radioactive waste, and accidents such as have happened in Chernobyl or Fukushima are not possible [14]. This makes fusion a safe and desirable source of energy.

In order for these reactions to be able to occur, a plasma needs to be created. To achieve this, massive amounts of energy are used to heat up a gas enough to become a plasma which is then confined in the reactor using a magnetic field. This environment needs to be maintained for long periods of time for fusion reactions to occur. Projects such as ITER [14] are currently trying to build devices where this can happen. However, this is complicated by microinstabilities inside the plasma that cause turbulences and with that, energy to be lost. In order for physicists to be able to counteract this problem, it is crucial to understand these instabilities better. This can be done by computer simulations that give information about the plasma's behaviour. For this purpose, we use the gyrokinetic solver Gene. The physical parameters that act as inputs for these simulations are subject to uncertainty. Therefore, an approach to these problems with a UQ framework is required. Increased studies of plasma microturbulence using UQ methodologies have only been conducted in the past several years which renders further investigation into this application necessary.

1.2. Overview over this work

This work examines multifidelity Monte Carlo sampling in detail. We study two test cases from the field of plasma microturbulence analysis: the so-called Cyclone Base Case with three input parameters and an extended version with an input dimension of eight. We show in practice how estimating mean and variance using MFMC estimators can improve the precision of standard Monte Carlo estimators by several orders of magnitude and present how MFMC can achieve this. As surrogate models for Gene, we use a sparse grid approximation and construct a neural network. Using these two

models, we discuss how combining multiple low-fidelity models can further increase the estimator accuracy.

1.3. Related work

In this section, we review some of the techniques used in existing works to quantify uncertainty in the analysis of plasma microinstabilities. We further summarize relevant literature on multifidelity methods in UQ and in particular, multifidelity Monte Carlo sampling.

The study of plasma microturbulence problems by UQ methods has been receiving increasing attention in recent years. Different methodologies have been applied to this type of scenario. The work [9] conducts a simple parameter scan with the goal of validating the solver GENE, which is also employed in our simulations, against a specific test case. Similar studies were carried out in [38]. The authors of [39] perform a UQ study on nonlinear plasma turbulence test cases using the probabilistic collocation method which improves on simple Monte Carlo sampling.

One important area of UQ is sensitivity analysis. This can be summarized as the quantification of the relative contributions of a model's uncertain input parameters to its output uncertainty [32]. In [40] a sensitivity analysis is performed, again with the objective of validating nonlinear simulations using a plasma microturbulence framework against data obtained from experiments. In the work [30], further study of the impact of input parameters on model output is presented using a spectral method.

In [8] multiple scenarios addressing plasma microinstabilities are analysed with a UQ approach. For this, an approximation of GENE is developed using adaptive sparse grids and a sensitivity scoring system. We make use of this approximation for our work as well (see Section 2.3).

A general introduction to multifidelity methods is given in [26]. The work proposes a classification of different types of surrogate models and model management techniques (see Subsection 4.1.3), and surveys multifidelity techniques used in UQ, statistical inference and optimization. In [22], multifidelity UQ is performed by constructing approximations of a model using stochastic collocation and polynomial chaos methods. Three different multifidelity techniques for UQ are discussed in [15], among them multifidelity Monte Carlo sampling (MFMC), the algorithm employed in the current work.

Control variates, the variance reduction technique that MFMC is based on, are summarized e.g. in [21]. In [24], this classical approach is extended to situations where the mean of the control variate is not known, as is the case for our scenario. [33] shows a means to judge the efficiency of a control variate estimator which we will also use in

our work. Control variates and resulting multifidelity Monte Carlo estimators for UQ are described in [23]. The work [25] shows how optimal results can be achieved with multifidelity Monte Carlo sampling for a given set of models. The authors formalize this approach and discuss how the employed models should be chosen. Moreover, [29] presents multifidelity Monte Carlo estimators for the variance of model output and for sensitivity indices which are again based on the control variate technique. The algorithms employed in this work are primarily used as described in the last two of the mentioned works. To the best of our knowledge, our work is the first application of multifidelity Monte Carlo sampling to plasma microturbulence analysis.

1.4. Outline of this work

The remainder of this thesis is structured as follows. The following three chapters introduce the theoretical aspects that this work is based on. Chapter 2 summarizes the real-world application of plasma microturbulence analysis that lies behind our experiments. It further gives information on the solver GENE and describes the construction of two surrogate models. In Chapter 3, the area of UQ that is relevant for this work, uncertainty propagation, is outlined in more detail. The chapter then presents definitions and notations from probability theory and statistics that are used throughout this work. Furthermore, a standard algorithm in uncertainty propagation is presented. Chapter 4 gives an overview over multifidelity methods in general and introduces multifidelity Monte Carlo sampling, the algorithm under consideration in this work. In Chapter 5 our results from the application of this technique to two plasma microturbulence test cases are presented. We give a conclusion and brief outlook in Chapter 6.

2. Plasma microturbulence analysis

This chapter gives an overview of the field of *plasma microturbulence analysis*. We choose this real-world application as a representative and practically relevant problem in which the impact of uncertainties needs to be properly analysed. Section 2.1 summarizes the physical background of this application. As a model for simulating plasma microinstabilities, we look at the simulation code GENE. An introduction to this code is given in Section 2.2. Sections 2.3 and 2.4 describe how to construct surrogate models for GENE. It should be noted that the full description of the physics of this application goes beyond the scope this work. For more detail regarding plasma microinstability analysis, we refer the reader to [16, 28].

2.1. Theoretical Background

Plasma microturbulence analysis is a highly relevant research area in the future construction of nuclear fusion reactors. Fusion power plants aim to generate power by harvesting the energy released in the fusion of particles. In order to make reactions like this possible, a *plasma* needs to be generated for the fusion to take place in. A plasma is essentially a gas that is brought to very high temperatures such that electrons separate from their nuclei and the gas becomes ionized. A key characteristic of a plasma is its low density. In order for high-energy particle collisions and therefore fusion reactions to occur, fusion devices need to confine the plasma with the help of a magnetic field. This field is designed to be of a toroidal shape and confines the plasma to that form [28]. Figure 2.1 shows a schematic of a tokamak, a fusion reactor in which the plasma is contained in this way. A current focus of research in the *ITER* project is the creation of a burning plasma, meaning a plasma that can sustain on its own the necessary temperature for fusion to occur [14].

An obstacle on the path to accomplishing this are plasma microinstabilities, i.e. fluctuations in the plasma that cause energy loss. The reason for which they occur are steep density and temperature gradients inside the plasma. These are unavoidable and lead to free energy that drives the turbulent transport. Strong magnetic fields can also not fully control these instabilities. For this reason, it is important for plasma physicists to better understand the microturbulences resulting from the instabilities and obtain information that can be used in the process of constructing such burning plasmas.

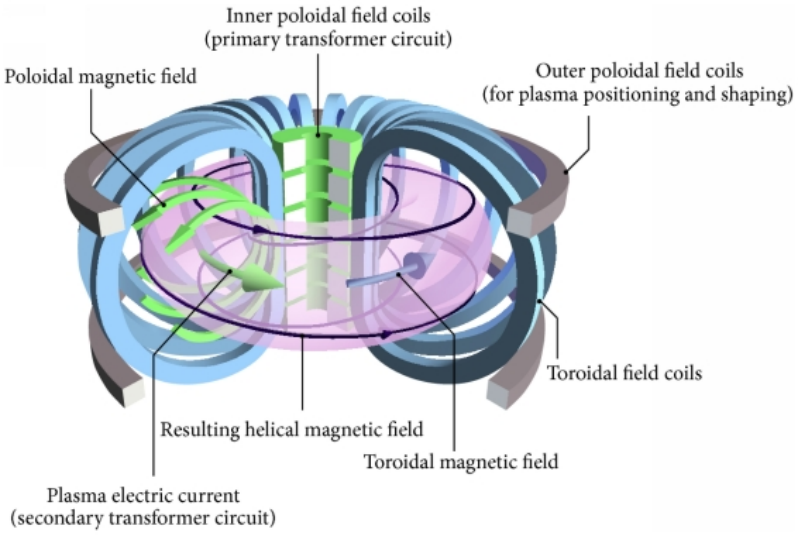


Figure 2.1.: Schematic of a tokamak fusion reactor. The plasma is confined by magnetic fields in a toroidal shape. Source: [18], licensed under CC BY 4.0.

To this aim, computer simulations of the plasma can be conducted. However, measuring the physical parameters which play a big role in the experiment, e.g. the temperature and density gradients, is not always possible with perfect precision. Therefore, these measurements are subject to uncertainties. Furthermore, plasma simulations can be stiff problems, meaning that small change in the inputs can have a big impact on the results [8]. This is why a UQ approach is needed in the analysis of plasma microturbulences.

In the following, we briefly summarize the modelling of fusion plasma. Already in fluid dynamics, which is a highly researched area of classical physics, dealing with turbulence is difficult. Moreover, since particle collisions in the plasma happen with a low-frequency, fluid descriptions are often not suitable. One approach to modelling plasma microinstabilities would be a kinetic model which uses 6D Vlasov equations per particle species coupled via Maxwell's equations. Solving these six-dimensional differential equations is very computationally expensive and therefore another approach with a reduced model needs to be used [8].

A popular technique that gives rise to computationally feasible models is that of *gyrokinetic theory*. For more detail on this than is given here, we refer to [4, 16]. Gyrokinetics gives a set of non-linear partial differential equations in 5D phase state that is computationally much more feasible to solve and is widely used in the modelling

of plasma dynamics. Particles in a plasma are known to circle around magnetic field lines at high speeds, while the exact point they are orbiting is slowly drifting [4]. The frequency at which the particles move around this so-called gyrocenter is by many orders of magnitude higher than the frequency at which turbulences usually occur [16]. For this reason, the gyrophase information of the particles can be omitted when modelling microinstabilities. This yields the mentioned system of five-dimensional equations. We will now look at a solver for these equations which we employ for our simulations.

2.2. The GENE code

GENE[1], which stands for *Gyrokinetic Electromagnetic Numerical Experiment*, is a numerical solver for the aforementioned gyrokinetic equations. It works on a 5D (3D-2V) position-velocity space plus time and uses a fixed grid [8]. Its development started in 1999 and is still ongoing [12]. GENE is massively parallelized in that it runs computations for different grid points on different cores. This is done using MPI [12].

GENE allows for different types of simulations, e.g. radially local and global experiments. Local simulations model flux-tube domains, meaning a narrow box that follows the field lines of the surrounding magnetic field. In these type of simulations, density and temperature profiles are only evaluated at the center of the flux-tube and periodic boundary conditions in directions perpendicular to the magnetic field lines are assumed. Computations for this type of simulations can be done efficiently using spectral methods [11]. Radially global scenarios extend the simulations to full torus simulations. This is necessary for e.g. small devices where locality can not be assumed. In these situations, periodic boundaries can not be used any more and computations become much more expensive and can require at least 10k CPU-hours [8].

Additionally, it is possible to run GENE for linear simulations, i.e. simulations which do not take into account the nonlinearities in the gyrokinetic equations. Operating GENE in a nonlinear mode can massively increase the runtime. For this reason, the test cases we examine in this work are linear local simulations. Linear simulations can be highly relevant in e.g. identifying different microinstabilities such as ITG or ETG driven modes, i.e. modes in which instabilities are driven by the temperature gradients of ions or electrons, or TEMs which are modes driven by trapped electrons [8].

Because GENE is still expensive to evaluate, a computationally cheaper surrogate can be used in its place. We summarize such a reduced model in the following.

[1] http://genecode.org/

2.3. Sparse grids surrogate

The sections presents an approach used to generate a surrogate model for GENE. This technique was first presented in [8]. Using an adaptive approach, it constructs an approximation of GENE via interpolation on sparse grids. The resulting model is then much less expensive to evaluate than GENE. For an introduction to sparse grids, we refer to [5]. In our experiments, we will use this surrogate as a low-fidelity model (see Subsection 4.1.1).

The goal in using sparse grids to construct this approximation is to delay the curse of dimensionality. The number of deterministic grid points needed for interpolation on a full uniform grid grows exponentially large with the number of stochastic parameters, i.e. the number of uncertain inputs. Using N points in one dimension, a full-grid approach requires N^d points in d input dimensions. For an increasing number of dimensions, this approach quickly stops being computationally feasible. Sparse grid methodologies target this problem by using significantly fewer grid points while maintaining a similar approximation quality as if a full grid had been used.

In the classical sparse grid approach, the grid points are typically chosen before execution of the algorithm. In contrast, adaptive sparse grid techniques only refine the grid in specific directions that are determined during the runtime of the algorithm. This information on whether to refine in a specific direction is obtained by estimating the approximation errors on the current grid [27].

The sparse grid approximation under consideration here uses such an adaptive approach, introducing a sensitivity scoring system which quantifies the impact the different stochastic parameters and their interaction have on the model output. The sensitivity scores refinement approach aims to preferentially refine the stochastic input directions that are rendered important.

In the following, we describe the idea behind the employed technique in more detail. The notations are used as presented in [8]. First, we summarize the employed sparse grid approach and then describe how the adaptivity is implemented. Let $\mathcal{U}^{(i)}$, $i = 1, \dots, d$ be a sequence of one-dimensional linear continuous operators. By d we denote the number of input parameters. Consider approximations $\mathcal{U}_\ell^{(i)}$ for $\mathcal{U}^{(i)}$ with

$$\|\mathcal{U}^{(i)} - \mathcal{U}_\ell^{(i)}\| \xrightarrow{\ell \to \infty} 0 \tag{2.1}$$

in a suitable norm $\|\cdot\|$. We define $\ell \in \mathbb{N}$ as the level of the approximation. The higher the level, the higher is the accuracy of the approximation $\mathcal{U}_\ell^{(i)}$. Note that sparse grid approximations can be defined for arbitrary linear continuous operators, such as interpolation, integration or spectral projection. In this work, we focus on interpolation because [8] established that this provides accurate approximations in the context of

plasma microturbulence at low cost.

We briefly summarize Lagrange interpolation as employed here. More detail can be found in [2]. We define $m(\ell)$ as the number of points used in constructing the approximation $\mathcal{U}_\ell^{(i)}$. The interpolation nodes on level ℓ are denoted by $\{u_n\}_{n=1}^{m(\ell)}$. With the Lagrange polynomials $\{L_n(u)\}_{n=1}^{m(\ell)}$ of degree $n-1$, we define $\mathcal{U}_\ell^{(i)}$ as the interpolation operator

$$\mathcal{U}_\ell^{(i)}(f)(u) = \sum_{n=1}^{m(\ell)} f(u_n) L_n(u). \tag{2.2}$$

In our case, the function f that is to be interpolated is GENE. Therefore, obtaining the necessary function evaluations is computationally expensive. For this reason, the line (L)-Leja points (see [10]) are chosen as interpolation nodes. This is because (L)-Leja points are nested, i.e. the points $\{u_n\}_{n=1}^{m(\ell-1)}$ are a subset of $\{u_n\}_{n=1}^{m(\ell)}$, meaning model evaluations from level $\ell-1$ can be reused for level ℓ. Further, only one additional point is added per level, i.e. $m(\ell) = \ell$, and therefore the number of necessary model evaluations only grows slowly. The line (L)-Leja points are constructed as

$$\begin{aligned} u_1 &= 0.5 \\ u_n &= \underset{u\in[0,1]}{\operatorname{argmax}} \left|\prod_{k=1}^{n-1}(u-u_k)\right|, \quad n \geq 2. \end{aligned} \tag{2.3}$$

In the case that (2.3) has multiple maximizers, we simply pick one of them. Here, we show the (L)-Leja sequence on $[0,1]$. This can easily be generalized to an arbitrary domain by scaling the points to that domain. Figure 2.2 shows a full grid and a sparse grid constructed with (L)-Leja points.

With the approximations $\mathcal{U}_\ell^{(i)}$, the operator $\mathcal{U}^{(i)}$ can now be written as

$$\mathcal{U}^{(i)} = \mathcal{U}_1^{(i)} + \sum_{l=2}^{\infty} \mathcal{U}_\ell^{(i)} - \mathcal{U}_{\ell-1}^{(i)}. \tag{2.4}$$

for $i = 1, \ldots, d$. We define the individual terms of this sum as $\Delta_1^{(i)} = \mathcal{U}_1^{(i)}$ and

$$\Delta_\ell^{(i)} = \mathcal{U}_\ell^{(i)} - \mathcal{U}_{\ell-1}^{(i)} \tag{2.5}$$

for $\ell \geq 2$ and $i = 1, \ldots, d$ and obtain for the operators

$$\mathcal{U}^{(i)} = \sum_{\ell=1}^{\infty} \Delta_\ell^{(i)}. \tag{2.6}$$

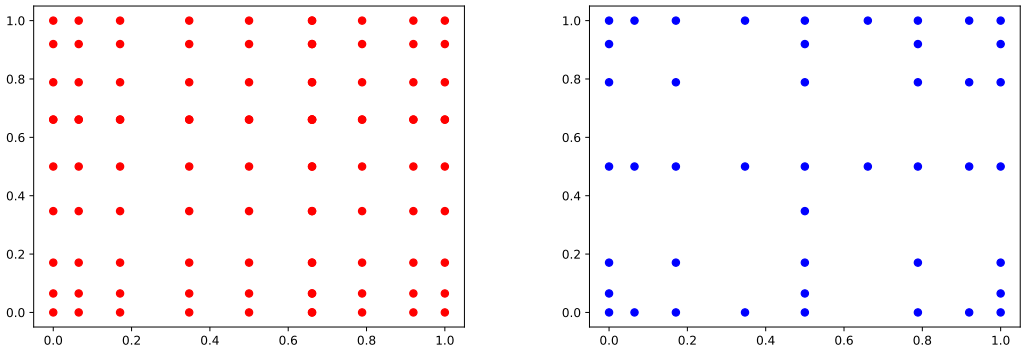


Figure 2.2.: Comparison of a full grid (left) and a sparse grid (right). Both are using line (L)-Leja points.

To extend these one-dimensional operators to d dimensions, we combine them via tensorization and have

$$\mathcal{U}^{(\mathbf{d})} = \mathcal{U}_{\ell_1}^{(1)} \otimes \ldots \otimes \mathcal{U}_{\ell_d}^{(d)} = \sum_{\boldsymbol{\ell}=0}^{\infty} \Delta_{\ell_1}^{(1)} \otimes \ldots \otimes \Delta_{\ell_d}^{(d)}. \tag{2.7}$$

Here, $\boldsymbol{\ell}$ is a multiindex with $\boldsymbol{\ell} = (\ell_1, \ldots, \ell_d) \in \mathbb{N}^d$ where ℓ_i signifies the level of refinement in dimension i. From this, an approximation for $\mathcal{U}^{(\mathbf{d})}$ can now be constructed. This is done by cutting off the summation over the multiindex $\boldsymbol{\ell}$ at a certain point. For this purpose, let $\mathcal{L} \subset \mathbb{N}^d$ be a finite multiindex set. An approximation only computes a sum over this set $\mathcal{L}$, i.e. stops the refinement for each direction at a different level. Figure 2.3 depicts the sparse grid built from a multiindex set $\mathcal{L}$. The approximation $\mathcal{U}_{\mathcal{L}}^{(\mathbf{d})}$ is given by

$$\mathcal{U}^{(\mathbf{d})} \approx \mathcal{U}_{\mathcal{L}}^{(\mathbf{d})} = \sum_{\boldsymbol{\ell} \in \mathcal{L}}^{\infty} \Delta_{\ell_1}^{(1)} \otimes \ldots \otimes \Delta_{\ell_d}^{(d)}. \tag{2.8}$$

The challenge then lies in constructing the multiindex set $\mathcal{L}$ as this determines how fine the grid is chosen in each dimension. This is done adaptively based on an error indicator for the current level of refinement. In contrast to many previous algorithms, the approach considered here uses local as opposed to global information. For this, a sensitivity scoring system is constructed which measures the contribution of each direction as well as the contribution from the interaction between all of the dimensions. The user prescribes a set of tolerances $\boldsymbol{\tau} = (tol_1, \ldots, tol_d, tol_{d+1})^T$. If the contribution of

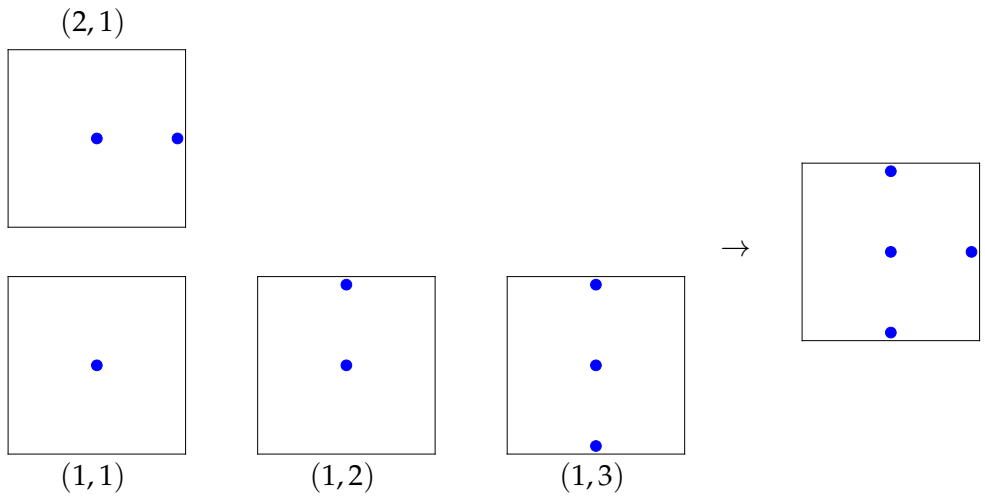


Figure 2.3.: The (L)-Leja grid points and the corresponding multiindices and the resulting sparse grid constructed from the set $\mathcal{L} = \{(1,1),(1,2),(1,3),(2,1)\}$.

parameter i at a certain level drops lower than tol_i, refinement of the grid is stopped in this direction. With tol_{d+1}, a threshold for the interaction between all of the input parameters is specified. In this way, the algorithm adapts to the structure of the problem at hand and exploits all directions up to the given precision.

When creating a surrogate model for GENE using this approach, the parameter τ can be used to control the accuracy of the resulting model. Imposing lower tolerances leads to the algorithm constructing finer grids and therefore more accurate models. Less precise models can be obtained by prescribing higher tolerances. These in turn are then less expensive to evaluate. This simple construction of models that have differing levels of correlation with GENE can be used in our application of multifidelity methods which rely on having models of different accuracies. A second type of surrogate model we use in our experiments is one obtained via machine learning. We give some detail on this in the following section.

2.4. Machine learning surrogate

One important characteristic of multifidelity Monte Carlo sampling is the fact that many different models can be used in addition to the high-fidelity model. MFMC is non-intrusive, meaning it essentially treats each model as a black box and only uses them to obtain evaluations at given samples. Therefore, it does not matter how the

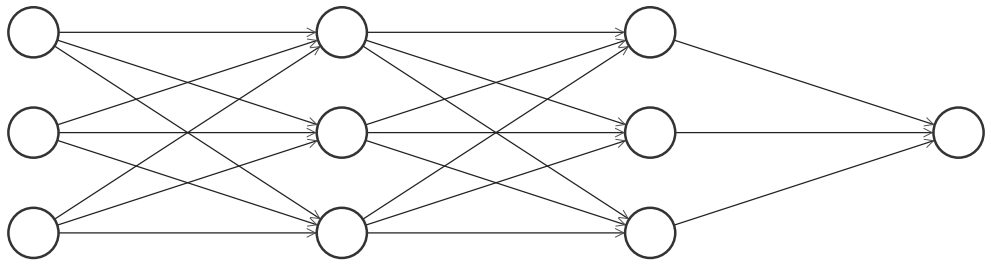


Figure 2.4.: Our neural network contains two hidden layers with three neurons per layer.

different models are constructed. Given that there are often multiple ways available in which physics applications can be modelled, this is very attractive.

In order to demonstrate this flexibility that MFMC offers, we construct another surrogate model for GENE. With output data from previous runs of GENE, we create a simple data-fit approximation with a machine-learning approach. For this, we use the neural-networks API Keras[2] running on Tensorflow with Python.

The problem solved here using a neural network is a regression problem. This means that we want the model to predict output that is a single continuous value from a given number of input parameters. The model aims to approximate a non-trivial function $f : \mathbb{R}^d \to \mathbb{R}$ (in our case, f is GENE) by a function $g : \mathbb{R}^d \to \mathbb{R}$ constructed from training data, that is pairs of input parameters $\mathbf{x} = (x_1, \ldots, x_d)^T$ and the corresponding model evaluation $f(\mathbf{x})$. Regression typically determines g using the following approach:

$$g(\mathbf{x}, \mathbf{w}) = w_0 + w_1\phi_1(\mathbf{x}) + \ldots + w_{M-1}\phi_{M-1}(\mathbf{x}). \tag{2.9}$$

By ϕ_i, $i = 1, \ldots, M-1$ we denote the so-called *basis functions* and $\mathbf{w} = (w_0, ..., w_{M-1})^T$ are parameters that are used to fit the function to the given data. By using non-linear functions, complex non-linear input-output relationships can be approximated. The number of parameters M determines the degrees of freedom in fitting g. Training a regression machine learning model then means determining the optimal values for the parameters w_i such that the approximation error of g is minimal. For more detail on regression and machine learning in general, we refer to [3].

The structure of the neural network we create here is illustrated in 2.4. The input layer of the machine-learning picture takes three input parameters. It has two densely connected hidden layers, both of which are made up of three neurons each. The output

[2] https://keras.io/

layer returns a single value. As an activation function, we use the commonly employed ReLU function. An RMSprop optimizer with a learning rate of 0.001 is chosen as an optimizer to use during training to determine the weights and biases of the neural network. This is an optimizer that improves on the often employed standard gradient descent method. We have found this choice to make training fast. This optimizer is then set up to minimize the MSE of the predicted output during training. We show how this can be accomplished using Keras in Section A.1.

3. Uncertainty Quantification

In Section 1 we gave a brief introduction to the discipline of uncertainty quantification. In this work we focus on one area of UQ, *forward propagation of uncertainty*, which will be described further in Section 3.1. Section 3.2 introduces notations and summarizes the aspects of probability theory that are necessary for understanding the methodology used in this work. In Section 3.3, we detail Monte Carlo sampling, which is a standard algorithm used in uncertainty propagation. In later chapters, we focus on multifidelity Monte Carlo sampling, which is based on this technique.

3.1. Forward Propagation of Uncertainty

In this section, we give more detail on the specific kind of UQ problem that arises with our plasma physics scenario. Typically, UQ problems can be categorized into two types: *forward propagation of uncertainty* (or forward uncertainty quantification), and *inverse uncertainty quantification*.

In forward UQ, the uncertainties of the system are modelled in the input parameters. This appears in applications where the exact values of inputs are not known, e.g. when the precision of physical measurements of input parameters is limited. This uncertainty is often represented as a probability distribution function for each of the input parameters. The objective is to learn about the effect these uncertainties have on the output of the employed model. This can be quantified by estimating the statistical moments of the model output, such as expectation and variance, based on the uncertain input parameters. To determine these moments, the input uncertainties need to be propagated through the model. The real-world application considered in this work is an example of forward propagation of uncertainty where, based on expert opinion, a certain probability distribution of the input parameters is assumed. The principle of forward UQ is illustrated in Figure 3.1 [34].

In contrast to this, inverse UQ is given model output and from it, aims to characterize uncertainties in the input parameters. As this work solely deals with a forward UQ problem, we do not go into more detail on inverse problems here. For more details on both of the mentioned areas of UQ, see [32].

It is also important to mention that forward propagation of uncertainty is an *outer loop application*. In outer-loop applications we construct a loop around a model f which

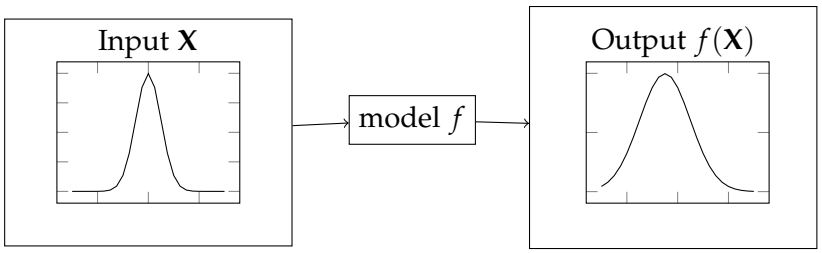


Figure 3.1.: Forward UQ estimates statistics of the model output based on the input distribution.

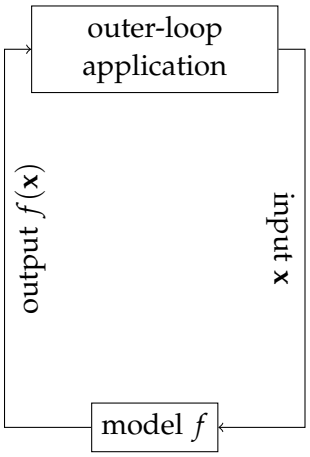


Figure 3.2.: Schematic of an outer-loop application. The outer-loop application uses the given model to compute independent results and aggregates them.

then is evaluated in each iteration for a realization $\mathbf{x}$ of the set of input parameters. At the end, the outputs $f(\mathbf{x})$ of each iteration are used to compute the desired statistics, see Figure 3.2 [26]. Often, these iterations are independent of each other and therefore can be run in parallel [26].

In order to understand the algorithms we present here that are used in forward UQ, we need some concepts from the fields of probability theory and statistics. We introduce those in the following.

3.2. Probabilistic Framework

In this section, an overview of the aspects of probability theory that are used in this work is given. Definitions and notations mainly follow [32, 19].

A *probability space* is given by $(\Omega, \mathcal{F}, \mathcal{P})$, where Ω is the sample space, i.e. the set of all possible outcomes, the set of events $\mathcal{F}$ is a σ-algebra on Ω which holds sets of possible outcomes, and $\mathcal{P}$ is a probability function $\mathcal{P} : \mathcal{F} \to [0,1]$. We define a *random variable* X as a function $X : \Omega \to \mathbb{R}$ that assigns a probability to each of the outcomes. In our case we only make use of continuous random variables, i. e. random variables whose image set is not countable.

A random variable's *cumulative distribution function* (cdf) $F_X : \mathbb{R} \to [0,1]$ is defined as $F_X(x) = \mathcal{P}[\{\omega \in \Omega | X(\omega) \leq x\}]$. For a continuous random variable X it can be written as

$$F_X(x) = \int_{-\infty}^{x} f_X(s)ds \tag{3.1}$$

where $f_X : \mathbb{R} \to [0,1]$ is the *probability density function* (pdf) of X with $f_X = \frac{dF_X}{dx}$.

The *expected value* or *mean* $\mu_X = \mathbb{E}[X]$ of a continuous random variable X is given by

$$\mathbb{E}[X] = \int_{\mathbb{R}} x f_X(x)dx. \tag{3.2}$$

For the *variance* $\sigma_X^2 = \mathrm{Var}[X]$ of a continuous random variable X it holds that

$$\mathrm{Var}[X] = \mathbb{E}[(X - \mu_X)^2] = \int_{\mathbb{R}} (x - \mu_X)^2 f_X(x)dx. \tag{3.3}$$

The *standard deviation* of a random variable X is denoted by σ_X and is given by the positive square root of the variance $\mathrm{Var}[X]$. The *covariance* of two random variables X and Y is given by

$$\mathrm{Cov}(X,Y) = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])]. \tag{3.4}$$

The *Pearson correlation coefficient* of two random variables X and Y is defined as

$$\rho_{XY} = \frac{\mathrm{Cov}(X,Y)}{\sigma_X \sigma_Y}. \tag{3.5}$$

It takes values between -1 and $+1$ and is a measure of linear correlation between the two random variables [31]. A correlation coefficient of 0 means that the random variables are uncorrelated.

Let $\mathbf{X} = (X_1, \ldots, X_d)$ be a *random vector* of d random variables. Their joint cdf $F_{\mathbf{X}} : \mathbb{R}^n \to [0,1]$ is given by $F_{\mathbf{X}}(x_1, \ldots, x_d) = \mathcal{P}[\{\omega \in \Omega | X_i(\omega) \leq x_i, i = 1, \ldots, n\}]$. In the remainder of this work, the notation $\mathbf{X}$ will always refer to a random vector

consisting of multiple random variables. We call $\mathbf{X}$ *independent* if, and if only, for $x_1, \ldots, x_d \in \mathbb{R}$ the joint pdf can be expressed as

$$f_X(x_1, \ldots, x_d) = \mathcal{P}(X_1 = x_1, \ldots, X_d = x_d) = \prod_{i=1}^{d} f_{X_i}(x_i). \tag{3.6}$$

Another important notion is that of *independent and identically distributed* (i.i.d.) random variables. Random variables $X_1, \ldots, X_d$ are i.i.d. if they are independent according to (3.6) and if their individual distribution functions are identical, i.e. $f_{X_1} = \ldots = f_{X_d}$.

In later chapters each one of d uncertain input parameters will be modelled as a random variable X_i for $1 \leq i \leq d$. We apply a model $f : \mathbb{R}^d \to \mathbb{R}$ to the input $\mathbf{X} = (X_1, \ldots, X_d)$ and obtain output $Y = f(\mathbf{X})$. Y is then again a random variable. In our experiments we assume independence of the input parameters.

The goal in this work is to estimate the mean $\mathbb{E}[Y]$ and the variance $\mathrm{Var}[Y]$. For this, a probability distribution of the input needs to be assumed. In the test cases analysed in this work each of the input parameters is modelled as a *uniform* random variable. The pdf of a random variable X that is uniformly distributed on an interval $[a, b]$ is given by $f_X(x) = \frac{1}{b-a}$ for $a \leq x \leq b$. We denote a uniform distribution by $X \sim \mathcal{U}(a, b)$.

Finally, the concept of *estimators* is introduced. We will use this all throughout this work. In practice, as is the case with our application, models are often much too complex for mean and variance to be computed analytically. This is only possible for very simple cases, such as linear models. For this reason, we have to estimate them numerically. An estimator is a random variable which maps (usually i.i.d.) random samples $\mathbf{X} = (X_1, \ldots, X_n)$ to an estimate for an unknown quantity of a probability distribution. If the unknown quantity is θ, then the estimator is often called $\hat{\theta}$. With this, an estimator can be written as a function $\hat{\theta} : \mathbb{R}^n \to \mathbb{R}$ and the resulting estimate is a realization $\hat{\theta}(\mathbf{X})$ of the estimator. To generate estimates for mean and variance of a random variable X we employ two very commonly used estimators. For samples $X_1, \ldots, X_n$ the *sample mean* and *variance* are given by

$$\hat{E} = \frac{1}{n} \sum_{i=1}^{n} X_i \quad \text{and} \quad \hat{V}^2 = \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \hat{E})^2. \tag{3.7}$$

Both of these estimators are *unbiased*. The *bias* of an estimator $\hat{\theta}$ is defined as the difference of the estimator's expected value to the true value of the estimated quantity θ, i. e. $Bias(\hat{\theta}) = \mathbb{E}[\hat{\theta}] - \theta$ [7]. If an estimator is unbiased, it means that the bias is zero and therefore its expected value is equal to that of the quantity it is trying to estimate. Thus, here it holds that $\mathbb{E}[\hat{E}] = \mu_X$ and $\mathbb{E}[\hat{V}^2] = \sigma_X^2$.

The *sample correlation coefficient* for two random variables X and Y approximates their Pearson correlation coefficient ρ_{XY}. Based on n i.i.d. samples (x_i, y_i) of the pair (X, Y)

it can be computed as

$$\bar{\rho}_{XY} = \frac{1}{\hat{V}_X \hat{V}_Y (n-1)} \sum_{i=1}^{n} (x_i - \hat{E}_X)(y_i - \hat{E}_Y) \tag{3.8}$$

where $\hat{V}_X^2$ and $\hat{V}_Y^2$ are the sample variances and $\hat{E}_X$ and $\hat{E}_Y$ are the sample means of X and Y as described in (3.7) [25]. With this, a standard method employed in forward UQ that makes use of the sample mean and variance is presented.

3.3. Monte Carlo Sampling

Monte Carlo methods are a very common set of algorithms used in different areas of mathematics, including statistics and UQ. First described in the 1940s by von Neumann, Ulam and Metropolis, Monte Carlo methods are based on evaluating a model for a number of random samples of the input and then computing an overall result from the simulation outputs [37]. A method that falls into this category is *Monte Carlo sampling*. Subsection 3.3.1 describes this algorithm and some of its attributes. Monte Carlo sampling is demonstrated on a simple example in Subsection 3.3.2.

3.3.1. Algorithm and properties

Monte Carlo sampling is a simple and straightforward method that is used to estimate statistical moments of a random variable. In the context of UQ, it can be used to obtain estimates for the mean and variance of a model output $Y = f(\mathbf{X})$ based on uncertain input $\mathbf{X}$. The necessary steps are shown in Algorithm 1.

Algorithm 1 Monte Carlo Sampling

Require: number of samples $n \in \mathbb{N}$

Draw n i.i.d. samples $\mathbf{x_i} = (x_{i1}, \ldots, x_{id}), i = 1, \ldots, n$ of the set of input parameters $\mathbf{X} = (X_1, \ldots, X_d)$ based on a given joint pdf $f_{\mathbf{X}}$.

for $1 \leq i \leq n$ **do**

Evaluate the model f for sample $\mathbf{x_i}$.

end for

Compute estimates for the quantity of interest, for our case the mean $\mathbb{E}[Y] = \mathbb{E}[f(\mathbf{X})]$ of the model output Y using the sample mean $\hat{E} = \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{x_i})$ and the variance $\mathrm{Var}[Y] = \mathrm{Var}[f(\mathbf{X})]$ using the sample variance $\hat{V} = \frac{1}{n-1} \sum_{i=1}^{n} (f(\mathbf{x_i}) - \hat{E})$.

It is interesting to note that for a fixed input the problem solved for each set of realizations of the input random variables in step 2 of Algorithm 1 becomes deterministic.

Moreover, both of the estimators employed in the third step are unbiased estimators, meaning that $\mathbb{E}[\hat{E}] = \mathbb{E}[Y]$ and $\mathbb{E}[\hat{V}] = \mathrm{Var}[Y]$ (see Section 3.2). By the Law of Large Numbers, they converge towards the true quantities $\mathbb{E}[Y]$ and $\mathrm{Var}[Y]$ for $n \to \infty$ where n is the number of samples [35].

We now take a look at the error introduced by using a Monte Carlo estimator $\hat{\theta}$ to approximate a quantity θ. The *mean squared error* (MSE) of $\hat{\theta}$ with respect to the true value θ is given by

$$MSE[\hat{\theta}] = \mathbb{E}[(\hat{\theta} - \theta)^2]. \tag{3.9}$$

If $\hat{\theta}$ is unbiased, as is true for the sample mean and variance, it follows that

$$MSE[\hat{\theta}] = \mathbb{E}[(\hat{\theta} - \theta)^2] = \mathbb{E}[(\hat{\theta} - \mathbb{E}[\hat{\theta}])^2] = \mathrm{Var}[\hat{\theta}]. \tag{3.10}$$

In the case that $\hat{\theta}$ is biased, meaning that $Bias(\hat{\theta}) = \mathbb{E}[\hat{\theta}] - \theta \neq 0$, we have

$$MSE[\hat{\theta}] = \mathbb{E}[(\hat{\theta} - \theta)^2] = \mathbb{E}[(\hat{\theta} - \mathbb{E}[\hat{\theta}] + Bias(\hat{\theta}))^2] = \mathrm{Var}[\hat{\theta}] + (Bias(\hat{\theta}))^2. \tag{3.11}$$

We see that the MSE of an estimator is directly influenced by its variance. Consider the sample mean $\hat{E}$ which is employed here. Based on the fact that it is unbiased, we can further give [23]

$$MSE[\hat{E}] = \mathrm{Var}[\hat{E}] = \frac{1}{N^2}\mathrm{Var}\left[\sum_{i=1}^{N} f(\mathbf{x_i})\right] = \frac{\mathrm{Var}[f(\mathbf{X})]}{N}. \tag{3.12}$$

With this, the convergence rate of the root mean squared error $\sqrt{MSE[\hat{E}]}$ of Monte Carlo sampling is given by $\mathcal{O}(\frac{1}{\sqrt{N}})$ where n is the number of samples [17]. Here it can be observed that the speed of convergence only depends on the number of samples and not on the dimensions of the input domain. This makes Monte Carlo sampling a powerful tool in solving higher dimensional problems for which the *curse of dimensionality* can quickly become an issue when other methods are used [26].

Looking again at (3.12), we note that in order to decrease the MSE of Monte Carlo sampling and therefore increasing the accuracy of the results, there are two possibilities. We can either increase the number of samples n, or we can try to decrease the variance $\mathrm{Var}[f(\mathbf{X})]$. In practice, the number of samples that are needed to obtain results of the desired accuracy is often prohibitive. This is especially the case if evaluating the employed model is computationally expensive and can therefore not be done for the required number of samples within realistic time constraints. Another approach is therefore decreasing the variance of the random variable whose mean is being estimated.

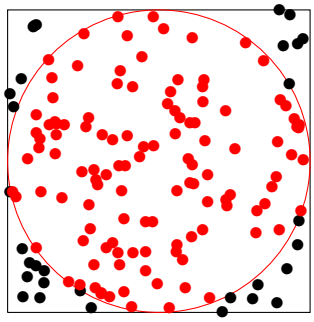


Figure 3.3.: To approximate π, draw uniform points inside the unit square and count the number of points n_{in} that lie inside the unit circle.

This is what *Multifidelity Monte Carlo Sampling* aims to accomplish. This technique which will be explained further in Chapter 4.

Before we demonstrate Monte Carlo sampling on an example, we give some more brief context on it. First and foremost, Monte Carlo sampling is a numerical integration method. Suppose, given a function $f : [a,b]^d \to \mathbb{R}$, we want to compute the value of the definite integral $I(f) = \int_{[a,b]^d} f(\mathbf{x}) d\mathbf{x}$ where $\mathbf{x} = (x_1, \ldots, x_d)$. While there exist many quadrature rules for the one-dimensional case, for $d > 1$ this is in general more complex. In particular, numerical integration is subject to the curse of dimensionality. An approach that uses n fixed grid points in one dimension requires n^d points in d dimensions. Monte Carlo integration on the other hand draws n uniform i.i.d. samples $\mathbf{x_i} \in [a,b]^d$ with $1 \leq i \leq n$ from the domain $[a,b]^d$ and approximates the integral as $\hat{I}(f) = V \cdot \frac{1}{N} \sum_{i=1}^{n} f(\mathbf{x_i})$, where $V = \int_{[a,b]^d} d\mathbf{x}$ is the volume of the domain $[a,b]^d$. As mentioned before, for $N \to \infty$, we have $\hat{I}(f) \to I(f)$ and in particular, no dependence on d which makes Monte Carlo sampling a popular method for quadrature in multiple dimensions. Comparing the approximation $\hat{I}(f)$ with the sample mean (3.7) we use to estimate a random variable's expectation, we see that with our estimators we are in fact approximating an integral and therefore solving an integration problem. For more details on Monte Carlo integration, please refer to [17].

3.3.2. Approximation of π

To demonstrate the functionality of Monte Carlo sampling, this subsection shows how it can be used to estimate the number π. This can be done via quadrature.

Since $r = 1$ for the unit circle, π can directly be approximated by estimating the area A of the circle, i. e. $A = \pi$. To do this, Monte Carlo sampling can be employed. The idea behind this application is as follows. We draw n sample points uniformly from

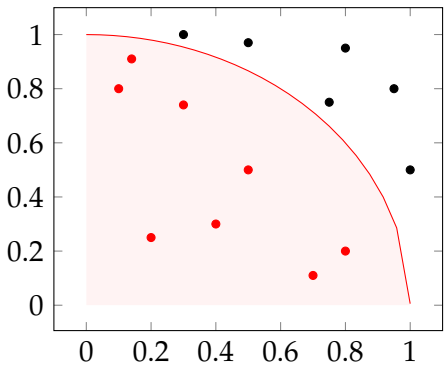


Figure 3.4.: A plot of the function $x_1^2 + x_2^2 = 1$. The function f from 3.13 equals 1 for all the points in red and 0 for the black points. The area coloured in red gives $\frac{1}{4}A$.

the unit square and count the number n_{in} of points drawn that lie in the unit circle as opposed to the number of those that lie outside it. With this, the area of the circle can be estimated as $\frac{n_{in}}{n}$. This idea is illustrated in Figure 3.3.

We express this more formally. Let $f : [0,1]^2 \to \{0,1\}$ with

$$f(x_1, x_2) = \begin{cases} 1 & \text{if } x_1^2 + x_2^2 \leq 1 \\ 0 & \text{otherwise.} \end{cases} \tag{3.13}$$

The function f therefore indicates whether a point $\mathbf{x} = (x_1, x_2)$ lies in the unit circle or not. Based on Algorithm 1, the steps shown in Algorithm 2 can now be used to approximate A and therefore π.

Algorithm 2 Approximation of π

Require: number of samples $n \in \mathbb{N}$
Draw n i.i.d. samples $\mathbf{x_i} = (x_1, x_2)$ of two uniform random variables from the domain $[0,1]^2$.
for $1 \leq i \leq n$ **do**
Evaluate $f(x_1, x_2)$.
end for
Calculate an estimate for the area A as $\hat{A} = 4 \cdot \frac{1}{n} \sum_{i=1}^{n} f(x_1, x_2)$.

As described in Subsection 3.3.1, the higher the number of samples n is chosen, the closer the estimate $\hat{A}$ will be to the true value of $\pi = 3.141592$. We show this convergence experimentally in Table 3.1.

number of samples n	**estimate for** π
100	3.2
1000	3.108
10000	3.1564
100000	3.14004
1000000	3.140128

Table 3.1.: Approximations of π obtained using the Algorithm 2 for an increasing number of samples.

4. Multifidelity Monte Carlo Sampling

In this chapter, we summarize the *Multifidelity Monte Carlo Sampling* (MFMC) algorithm, which is the main algorithmic focus of this work As explained in Chapter 3, estimating statistics of a model output using standard Monte Carlo sampling (MC) is not always computationally feasible. To obtain accurate results, one often needs to evaluate the model for more samples than possible. This is in part due to the employed models being expensive to evaluate. For the same computational costs as standard MC, MFMC sampling aims to produce more accurate estimates. In other words, MFMC yields the same accuracy as standard MC for a smaller computational cost.

In Section 4.1 a general introduction to multifidelity methods is given and their functioning and characteristics are described. The origin of the employed MFMC estimator are control variates which are described in Section 4.2. The MFMC estimator is then discussed in Section 4.3. In Section 4.4 we show how to minimize the MSE of this estimator. The employed algorithm for multifidelity Monte Carlo sampling is summarized in Section 4.5. Finally, we demonstrate the benefit of using a multifidelity estimator on a small numerical example in Section 4.6 and compare the results to those obtained by standard Monte Carlo sampling. For more details on the methods presented in this chapter please refer to [26, 25, 29].

4.1. Introduction to Multifidelity Methods

Before the estimators used to obtain the results in this work are presented, we first overview over multifidelity methods and the idea behind them. Because they can be used in many different fields of applied mathematics, we keep this section general and go into detail on their application as MFMC later. In the following, the definitions and notations are similar to those given in [26].

4.1.1. High- and low-fidelity models

The core idea of multifidelity methods is to use at the same time all available models characterizing a real-world phenomenon. Let $\mathcal{X} \subseteq \mathbb{R}^d$ be the input domain for both models and $\mathcal{Y} \subseteq \mathbb{R}^{d'}$ be the output domain with dimensions $d, d' \in \mathbb{N}$. We differentiate between a *high-fidelity model* and *low-fidelity model*. The high-fidelity model $f^{(0)} : \mathcal{X} \to \mathcal{Y}$

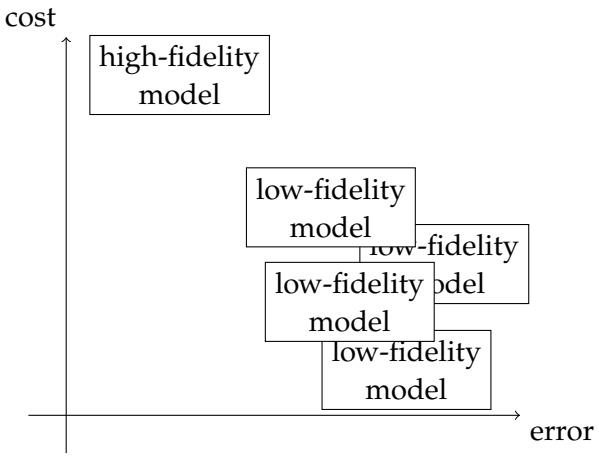


Figure 4.1.: Multifidelity methods use high- and low-fidelity models. The high-fidelity model is typically expensive to evaluate but high in accuracy, while the low-fidelity models are computationally cheaper and exhibit higher errors.

depicts reality with a degree of accuracy that is sufficient for our application. The low-fidelity models $f^{(i)} : \mathcal{X} \to \mathcal{Y}$ for $i = 1, \ldots, k$ with $k \in \mathbb{N}$ approximate the high-fidelity model but their output is of a lower accuracy. Usually the computational costs for evaluating the high-fidelity model are higher than the costs for evaluating the low-fidelity models. This relationship is visualized in Figure 4.1 [26].

In our case, GENE (see Section 2.2) acts as the high-fidelity model. We assume its output to be our "truth", meaning that its accuracy is sufficient for our purposes. As low-fidelity models we use a sparse grid approximation of GENE and a machine-learning model.

A multifidelity method uses evaluations from both the high- and low-fidelity models to obtain an overall solution to a given problem, such as estimating statistics of the high-fidelity model. Typically, this is done to reduce the computational cost of model evaluations while maintaining the accuracy of the result as if only the high-fidelity model had been used. By distributing some of the work to the low-fidelity models, their lower runtime is exploited to speed up the computational process. The high-fidelity model is kept in the loop for accuracy guarantees, i.e., to keep the estimators unbiased. In contrast to this, traditional *model reduction* techniques replace the high-fidelity model with a lower fidelity surrogate which is then used in computations. This reduces the runtime of Monte Carlo sampling or other uncertainty propagation methods. However, to guarantee accurate estimates, the lower fidelity surrogate has to be certified, i.e., its approximation error with respect to the high-fidelity model has to be known.

The goal we achieve in our work by using multifidelity methods is to reduce the time

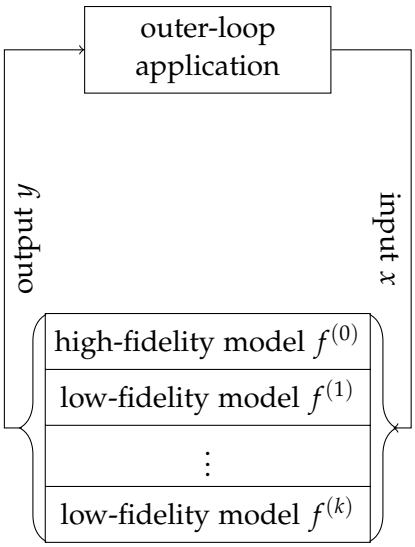


Figure 4.2.: The outer-loop application uses high-fidelity and one or more low-fidelity models to obtain a result.

budget p needed to generate estimates for the model output's mean and variance while maintaining the approximation quality of the results obtained by employing standard Monte Carlo sampling. This is equivalent to finding an estimator that has a lower MSE than the standard Monte Carlo estimator using the same given time budget.

Multifidelity methods can be used to obtain computational speedups in the solution of *outer-loop applications* which previously have solely used the high-fidelity model. We have defined the concept of outer-loop applications in the context of uncertainty propagation. Other scenarios that can be modelled as outer-loop applications include solving an optimization problem or dealing with inverse problems via statistical inference [26]. The concept of outer-loop applications using a high-fidelity and multiple low-fidelity models is illustrated in Figure 4.2 [26].

4.1.2. Types of low-fidelity models

To show the flexibility of multifidelity methods, it is important to mention the fact that many different types of low-fidelity models can be used in conjunction with the existing high-fidelity model. There are various ways to construct low-fidelity models which depend strongly on the representation in which the high-fidelity model is given. For example, it might be available only as a *black box*, meaning it can be used to compute output from given input but there is no insight into how these results are obtained. In another case, the implementation details of the high-fidelity model might be available

and well-understood. Again, we cite [26] where low-fidelity models are separated into three categories, see also [1] for more details:

1. *Simplified low-fidelity models*: This type of model leverages knowledge about the problem's physical background and/or details of the high-fidelity model's functionality to create a low-fidelity model. This can include simplifying the methods used to model real-world applications, using coarser grids to discretize a domain, or omitting non-linear terms from an equation.

2. *Projection-based low-fidelity models*: Instead of taking advantage of insight into the problem domain, projection-based models make use of the high-fidelity model's mathematical structure. In developing a low-fidelity model this structure is retained by finding a lower-dimensional subspace and projecting the mathematical operators used onto it.

3. *Data-fit low-fidelity models*: These are constructed from given input parameters and corresponding output data that has been obtained during multiple runs of the high-fidelity model. This is typically done via interpolation or regression. Machine-learning techniques can also be used to create a data-driven model. Note that creating this type of model is typically non-intrusive, meaning it does not need access to the high-fidelity model's implementation.

Both the sparse grid surrogate and the machine learning model we use as low-fidelity models are data-fit models, though both are constructed in very different ways.

4.1.3. Model management strategies

In order to balance the model evaluations of multifidelity methods between the high-fidelity and one or more low-fidelity models, a *model management strategy* is required, i.e. a rule for how to split the evaluations between the models and how to combine the separate results into a final result to the given problem. The authors of [26] give three different classes of model management strategies:

1. *Adaption*: The behaviour of the low-fidelity model is corrected with information from high-fidelity evaluations. Here, modifications to the low-fidelity model itself are made.

2. *Fusion*: This strategy combines evaluations from both types of models according to some algorithm to find a result.

3. *Filtering*: In this technique, information obtained from low-fidelity evaluations acts as a filter for when to invoke the high-fidelity model.

Fusion and filtering are both non-intrusive techniques. Multifidelity Monte Carlo uses a fusion model management strategy to accumulate the results of the different models. This technique is based on control variates which we introduce in the following.

4.2. Control Variates

In Subsection 4.1.3 we explained how operating multifidelity methods requires a model management strategy. For this purpose, the multifidelity estimator employed here uses *control variates*, which are a commonly used *variance reduction technique*. The goal of such techniques is to increase the precision of a MC estimator by decreasing its variance. Other variance reduction techniques include stratified sampling, importance sampling and antithetic variates [26, 13]. This section introduces control variates in a general sense and Section 4.3 then specifies their use in MFMC. For more details on control variates, see e.g. [21].

To see the need for such an approach to Monte Carlo sampling, consider again the error of a standard Monte Carlo estimator $\hat{E}$ for the mean as introduced in 3.7. Since $\hat{E}$ is unbiased, the MSE of this estimator is given by

$$MSE[\hat{E}] = \mathrm{Var}[\hat{E}] = \frac{\mathrm{Var}[f^{(0)}(X)]}{n} \tag{4.1}$$

where n is the number of samples (see 3.12). Decreasing the error of the estimator is therefore equal to decreasing its variance. As discussed earlier, using a higher number of samples is not always possible. Another approach is to decrease the term $\mathrm{Var}[f^{(0)}(X)]$. This can be achieved with control variates.

To reduce the estimator variance of a random variable A, control variates make use of one or more auxiliary random variables B that are correlated with A, the key being that the mean of these auxiliary variables is known, and further, that they are statistically correlated with A. It is shown later why this is necessary.

Consider estimating the expectation $\mathbb{E}[A]$ of a random variable A. Drawing n i.i.d. realizations a_i, $i = 1, \ldots, n$ of the random variable, this can be done via a standard Monte Carlo estimator

$$\bar{a}_n = \frac{1}{n} \sum_{i=1}^{n} a_i. \tag{4.2}$$

We know this estimator to be unbiased. Let B be a second random variable for which the expectation $\mathbb{E}[B] = \mu_B$ is known but can also be approximated from n i.i.d. samples

b_i, $i = 1, \dots, n$ of B using a Monte Carlo estimator

$$\bar{b}_n = \frac{1}{n} \sum_{i=1}^{n} b_i. \tag{4.3}$$

With this, we can give the control variate estimator $\hat{E}_A$ for $\mathbb{E}[A]$:

$$\hat{E}_A = \bar{a}_n + \alpha(\mu_B - \bar{b}_n) \tag{4.4}$$

where α is a real-valued constant. The term $\alpha(\mu_B - \bar{b}_n)$ represents an adjustment or correction to the standard Monte Carlo estimator $\bar{a}_n$. Since $\mathbb{E}[\bar{b}_n] = \mu_B$, it follows with the linearity of expectation that

$$\begin{aligned} \mathbb{E}[\hat{E}_A] &= \mathbb{E}[\bar{a}_n + \alpha(\mu_B - \bar{b}_n)] \\ &= \mathbb{E}[\bar{a}_n] + \alpha\mathbb{E}[\mu_B - \bar{b}_n] \\ &= \mathbb{E}[\bar{a}_n] \\ &= \mathbb{E}[A]. \end{aligned} \tag{4.5}$$

Thus, $\hat{E}_A$ is also unbiased just as the standard MC estimator. Furthermore, the work [23] shows the variance to be

$$\mathrm{Var}[\hat{E}_A] = \frac{1}{n}(\sigma_A^2 + \alpha^2\sigma_B^2 - 2\alpha\rho_{AB}\sigma_A\sigma_B). \tag{4.6}$$

Since $\hat{E}_A$ is unbiased, by (3.10) the MSE of the estimator is equal to its variance. Therefore minimizing the MSE of the estimator is again equivalent to minimizing its variance. By [21], this can achieved by choosing

$$\alpha_* = \frac{\mathrm{Cov}[A, B]}{\sigma_B^2} = \rho_{AB}\frac{\sigma_A}{\sigma_B} \tag{4.7}$$

and substituting into (4.6) leads to

$$MSE[\hat{E}_A^*] = \mathrm{Var}[\hat{E}_A] = (1 - \rho_{AB}^2)\frac{\sigma_A^2}{n} = (1 - \rho_{AB}^2)MSE[\bar{a}_n]. \tag{4.8}$$

Therefore, if A and B are correlated, i.e. $\rho_{AB} \neq 0$, the MSE of the control variate estimator is lower than that of the standard Monte Carlo estimator since $|\rho_{AB}| \leq 1$. Furthermore, the higher the correlation coefficient ρ_{AB} is, the higher is the variance reduction for a given number of samples n. Multifidelity Monte Carlo sampling uses this method to construct an estimator with a lower variance and makes use of low-fidelity models in place of the auxiliary random variable B. We specify this technique in the following.

4.3. Multifidelity Estimator

We now formulate the Multifidelity Monte Carlo estimators used to approximate $\mathbb{E}[f^{(0)}(\mathbf{X})]$ and $\mathrm{Var}[f^{(0)}(\mathbf{X})]$. They were first introduced in [25, 29] and are making use of the control variate technique as described above. While we have only shown control variates for one auxiliary random variable, the MFMC estimator here generalizes this technique to multiple random variables. We first describe how these estimators are computed in Subsection 4.3.1 and examine some of their attributes, such as their errors, in Subsection 4.3.2.

4.3.1. Definition

Here we compute statistics for the random variable $f^{(0)}(\mathbf{X})$, thus this takes the place of A from Section 4.2. As auxiliary random variables we use the outputs $f^{(1)}(\mathbf{X}), \ldots, f^{(k)}(\mathbf{X})$ of the k low-fidelity models. The multifidelity estimators are constructed as follows.

We define m_0 as the number of times the high-fidelity model $f^{(0)}$ is evaluated and $m_1, \ldots, m_k$ with $0 < m_0 \leq m_1 \leq \ldots \leq m_k$ as the number of model evaluations for the low-fidelity models $f^{(1)}, \ldots, f^{(k)}$. Further, m_k i.i.d. realizations

$$\mathbf{x_1}, ..., \mathbf{x_{m_k}} \tag{4.9}$$

of the input random vector $\mathbf{X}$ are drawn. We evaluate models $f^{(i)}$ for $i = 0, \ldots, k$ for the samples $\mathbf{x_1}, \ldots, \mathbf{x_{m_i}}$ and obtain

$$f^{(i)}(\mathbf{x_1}), ..., f^{(i)}(\mathbf{x_{m_i}}). \tag{4.10}$$

From these evaluations standard Monte Carlo estimators are derived. For the high-fidelity model $f^{(0)}$ we construct

$$\bar{E}^{(0)}_{m_0} = \frac{1}{m_0} \sum_{j=1}^{m_0} f^{(0)}(\mathbf{x_j}) \quad \text{and} \quad \bar{V}^{(0)}_{m_0} = \frac{1}{m_0 - 1} \sum_{j=1}^{m_0} (f^{(0)}(\mathbf{x_j}) - \bar{E}^{(0)}_{m_0})^2 \tag{4.11}$$

using the sample mean and variance, and for the low-fidelity models $f^{(i)}$, $i = 1, \ldots, k$ we compute

$$\bar{E}^{(i)}_{m_i} = \frac{1}{m_i} \sum_{j=1}^{m_i} f^{(i)}(\mathbf{x_j}) \quad \text{and} \quad \bar{V}^{(i)}_{m_i} = \frac{1}{m_i - 1} \sum_{j=1}^{m_i} (f^{(i)}(\mathbf{x_j}) - \bar{E}^{(i)}_{m_i})^2 \tag{4.12}$$

and additionally

$$\bar{E}^{(i)}_{m_{i-1}} = \frac{1}{m_{i-1}} \sum_{j=1}^{m_{i-1}} f^{(i)}(\mathbf{x_j}) \quad \text{and} \quad \bar{V}^{(i)}_{m_{i-1}} = \frac{1}{m_{i-1} - 1} \sum_{j=1}^{m_{i-1}} (f^{(i)}(\mathbf{x_j}) - \bar{E}^{(i)}_{m_{i-1}})^2. \tag{4.13}$$

It is important to see that while both of the estimators in (4.12) and (4.13) give estimates for the expectation $\mathbb{E}[f^{(i)}]$, the first uses m_i and the second m_{i-1} samples to do this. Since the m_{i-1} samples are a subset of the m_i samples, these estimators are dependent.

The multifidelity Monte Carlo estimator $\hat{E}$ of $\mathbb{E}[f^{(0)}]$ is then given by

$$\hat{E} = \bar{E}^{(0)}_{m_0} + \sum_{i=1}^{k} \alpha_i (\bar{E}^{(i)}_{m_i} - \bar{E}^{(i)}_{m_{i-1}}) \tag{4.14}$$

and the estimator $\hat{V}$ of $\mathrm{Var}[f^{(0)}]$ by

$$\hat{V} = \bar{V}^{(0)}_{m_0} + \sum_{i=1}^{k} \alpha_i (\bar{V}^{(i)}_{m_i} - \bar{V}^{(i)}_{m_{i-1}}) \tag{4.15}$$

where $\alpha_1, \ldots, \alpha_k \in \mathbb{R}$ are the control variate coefficients. It is specified how they should be chosen in Subsection 4.4.1. These estimators use the differences between $\bar{E}^{(i)}_{m_i}$ and $\bar{E}^{(i)}_{m_{i-1}}$ (or respectively $\bar{V}^{(i)}_{m_i}$ and $\bar{V}^{(i)}_{m_{i-1}}$) as a correction to the estimate obtained solely from high-fidelity evaluations. As mentioned in Subsection 4.1.3, the model management strategy employed here is that of fusion. Note also that the control variate estimator requires the true mean or variance of the auxiliary random variables (see (4.4)) which are not known in this case. We approximate them by $\bar{E}^{(i)}_{m_i}$ and $\bar{V}^{(i)}_{m_i}$ with $m_i > m_{i-1}$.

4.3.2. Properties

This subsection examines the given MFMC estimators (4.14) and (4.15) more closely. Firstly, it is important to state that both of the estimators are unbiased. With the same reasoning as in Section 4.2 it follows from the linearity of expectation that $\mathbb{E}[\hat{E}] = \mathbb{E}[f^{(0)}(\mathbf{X})]$ and $\mathbb{E}[\hat{V}] = \mathrm{Var}[f^{(0)}(\mathbf{X})]$.

Let $\mathbf{w} = (w_0, w_1, \ldots, w_k)^T$ with w_i, $i = 0, \ldots, k$, being the costs to evaluate model $f^{(i)}$ for one sample, and $\mathbf{m} = (m_0, m_1, \ldots, m_k)^T$ with m_i, $i = 0, \ldots, k$ being the number of model evaluations of model $f^{(i)}$. To obtain the correction terms, we evaluate model $f^{(i)}$ for the samples $\mathbf{x_1}, \ldots, \mathbf{x_{m_i}}$ to compute $\bar{E}^{(i)}_{m_i}$ and reuse the evaluations at the first m_{i-1} samples $f^{(i)}(\mathbf{x_1}), \ldots, f^{(i)}(\mathbf{x_{m_{i-1}}})$ to compute $\bar{E}^{(i)}_{m_{i-1}}$. Therefore, the computational costs $c(\hat{s})$ for constructing a multifidelity estimator $\hat{E}$ are

$$c(\hat{E}) = \sum_{i=0}^{k} w_i m_i = \mathbf{w}^T \mathbf{m}. \tag{4.16}$$

A multifidelity estimator $\hat{V}$ for the variance is obtained from the same samples as a mean estimator, therefore does not incur any additional costs. Because the objective

in constructing a multifidelity estimator was to reduce the variance of the resulting estimator, we examine this variance next. Note that since $\hat{s}$ is unbiased, the MSE of the estimator is again equal to its variance. In [25], the variance $\mathrm{Var}[\hat{E}] = MSE[\hat{E}]$ of the estimator for the expectation is shown to be

$$\mathrm{Var}[\hat{E}] = \frac{\sigma_0^2}{m_0} + \sum_{i=1}^{k} \left(\frac{1}{m_{i-1}} - \frac{1}{m_i}\right)\left(\alpha_i^2 \sigma_i^2 - 2\alpha_i \rho_{0,i} \sigma_0 \sigma_i\right), \tag{4.17}$$

where $\rho_{i,j}$ is the correlation coefficient between model outputs $f^{(i)}(\mathbf{X})$ and $f^{(j)}(\mathbf{X})$, and σ_i^2 is the variance of $f^{(i)}(\mathbf{X})$.

For the variance of the multifidelity estimator $\hat{V}$ we refer to the original publication [29]. In further computations, we will only make use of the above variance of the mean estimator. We explain the reason behind this in the following section.

The variance $\mathrm{Var}[\hat{E}]$ depends on the number of model evaluations $m_0, \ldots, m_k$ and on the control variate coefficients $\alpha_1, \ldots, \alpha_k$. Since their values directly influence the MSE of the resulting estimator, they should be chosen such that this error is as small as possible if an optimal MFMC estimator is to be obtained. This is addressed in the following section.

4.4. Optimal Model Managment

This section outlines the model management strategy used in constructing the multifidelity estimator described in Section 4.3. This strategy includes how to distribute the model evaluations to the different models, i.e. how to chose the m_i, and how to select the coefficients α_i. This is described in Subsection 4.4.1. It is shown that in order to minimize the error of the MFMC estimators, the employed low-fidelity models must be chosen with regard to their interaction with each other. Subsection 4.4.2 details how this can be best achieved.

4.4.1. Optimal number of evaluations and control variate coefficients

So far, we have defined the MFMC estimators $\hat{E}$ and $\hat{V}$ and explained that the model evaluations are distributed among the models according to $m_0, ..., m_k$. This section describes how to choose these numbers and it also shows the optimal values for the coefficients $\alpha_1, \ldots, \alpha_k$ such as to obtain an estimator with a minimal MSE.

We formulate this as an optimization problem for a given computational budget $p \in \mathbb{R}^+$. Since $MSE[\hat{E}] = \mathrm{Var}[\hat{E}]$ the variance from (4.17) is used as the objective function. Note that by using this function, the MSE of the mean estimator $\hat{E}$ is

minimized, not necessarily the MSE of the estimator $\hat{V}$. We describe reasons for this choice later in this subsection.

The optimal number of model evaluations $\mathbf{m}^* = (m_0^*, m_1^*, ..., m_k^*)^T$ and control variate coefficients $\alpha_1^*, \ldots, \alpha_k^*$ are thus obtained by solving the following optimization problem:

$$\begin{aligned} \underset{\substack{m_0,m_1,\ldots,m_k \in \mathbb{R} \\ \alpha_1,\ldots,\alpha_k \in \mathbb{R}}}{\operatorname{argmin}} \quad & \operatorname{Var}[\hat{E}] \\ \text{s.t. } & m_0 > 0, \\ & m_i \leq m_{i-1} \quad i = 1, \ldots, k, \\ & \mathbf{w}^T \mathbf{m} = p. \end{aligned} \tag{4.18}$$

The first two inequality constraints ensure that $0 < m_0^* \leq m_1^* \leq \ldots \leq m_k^*$ and the last condition guarantees that the cost of computing the MFMC estimator is equal to the time budget p. Note that here we only require $m_0, m_1, ..., m_k$ to be rational numbers, even though the number of model evaluations clearly need to integer. This is done in order to avoid having to solve complex integer programming problem and a relaxed linear program over the real numbers is solved instead. We round up the obtained m_i to $\lfloor \mathbf{m}^* \rfloor = (\lfloor m_0^* \rfloor, \lfloor m_1^* \rfloor, ..., \lfloor m_k^* \rfloor)^T$ to ensure that each model is evaluated at least once. The authors of [25] also reason that compared to a usually large number of model evaluations, rounding by a value less than one does not change the overall number significantly. In the following, we will assume the m_i^* to have been obtained by rounding and therefore, for them to be integer.

It is important to mention that the budget p should be chosen large enough such as to allow at least one evaluation of the high-fidelity model, that is, p should exceed the runtime needed for one high-fidelity model evaluation. If this is not given and only low-fidelity evaluations are used, the resulting estimator will be biased. The true values of $\mathbb{E}[f^{(0)}]$ and $\operatorname{Var}[f^{(0)}]$ can not be obtained from biased low-fidelity models alone.

We now give the unique solution to the optimization problem (4.18), i.e. the values $\mathbf{m}^*$ and $\alpha_1, ..., \alpha_k$ under which the largest variance reduction, and therefore the lowest MSE, can be achieved. For a proof of this solution, see [25]. In order for it to be correct, the models $f^{(0)}, f^{(1)}, \ldots, f^{(k)}$ need to be ordered according to their correlation coefficients:

$$|\rho_{0,0}| > |\rho_{0,1}| > \ldots > |\rho_{0,k}|. \tag{4.19}$$

Furthermore, their costs $w_0, w_1, \ldots, w_k$ need to fulfill

$$\frac{w_{i-1}}{w_i} > \frac{\rho_{0,i-1}^2 - \rho_{0,i}^2}{\rho_{0,i}^2 - \rho_{0,i+1}^2} \tag{4.20}$$

for $i = 1, \dots, k$. We define $\rho_{0,k+1}$ to be 0. The optimal coefficients $\alpha_1^*, \dots, \alpha_k^* \in \mathbb{R}$ are given by

$$\alpha_i^* = \frac{\rho_{0,i}\sigma_0}{\sigma_i} \tag{4.21}$$

for $i = 1, \dots, k$. To obtain the optimal number of model evaluations we first compute $\mathbf{r}^* = (r_0^*, r_1^*, \dots, r_k^*)^T \in \mathbb{R}_+^k$ by

$$r_i^* = \sqrt{\frac{w_0(\rho_{0,i}^2 - \rho_{0,i+1}^2)}{w_i(1 - \rho_{0,1}^2)}} \tag{4.22}$$

for $i = 0, \dots, k$ and then set

$$m_0^* = \frac{p}{\mathbf{w}^T\mathbf{r}^*} \quad \text{and} \quad m_i^* = m_1^* r_i^* \tag{4.23}$$

for $i = 1, \dots, k$. Choosing the control variate coefficients and the number of model evaluations according to these rules guarantees that the MSE of the MFMC estimator is minimized.

To be able to obtain the ordering (4.19), check condition (4.20) and compute the control variate coefficients and number of model evaluations, the correlation coefficients $\rho_{0,i}$ and the variances σ_i^2 are required for the high-fidelity and low-fidelity models. These are generally not known and must be obtained beforehand. We show in Subsection 4.5.2 how this can be achieved.

The optimization problem solved in this section minimizes the variance $\mathrm{Var}[\hat{E}]$, i.e. the variance of the multifidelity mean estimator. As recommended in [29], we will use the obtained coefficients and number of model evaluations for computing the MFMC variance estimator $\hat{V}$ as well. These may not be the optimal numbers to minimize $\mathrm{Var}[\hat{V}]$, but the work [29] shows that the MSE of the estimator that uses them is very similar to that of esimators using coefficients that minimize $\mathrm{Var}[\hat{V}]$. Furthermore, the term for the variance of the estimator $\hat{V}$ depends on the squared variances $(\sigma_i^2)^2$ and is therefore sensitive to changes in them. Since these variances are usually determined experimentally, this introduces a source of instability into the algorithm which can be avoided by using the coefficients optimized with regard to $\mathrm{Var}[\hat{E}]$ [29]. This is why, in the following sections, we will utilise these coefficients and focus on the MSE generated for the estimator $\hat{E}$.

Having obtained the coefficients $\alpha_1, \dots, \alpha_k$ and the number of model evaluations $m_0, m_1, \dots, m_k$ which minimize the MSE of the estimator $\hat{E}$, we can now examine this error more closely. The MSE of the MFMC estimator $\hat{E}$ with respect to a given time

budget p is given by [25]

$$MSE[\hat{E}] = \mathrm{Var}[\hat{E}] = \frac{\sigma_0^2}{p}\left(\sum_{i=0}^{k}\sqrt{w_i(\rho_{0,i}-\rho_{0,i+1}))}\right)^2 \tag{4.24}$$

where again $\rho_{0,k+1}$ is defined to be 0. To see that by using a MFMC estimator, we can achieve a reduction in the MSE compared to a standard Monte Carlo estimator, we compare the two errors. In (3.12), the MSE of a standard MC estimator $\hat{E}_{MC}$ for the mean of the high-fidelity model output $f^{(0)}(\mathbf{X})$ for a computational budget p is given as

$$MSE[\hat{E}_{MC}] = \frac{\sigma_0^2}{n} = \frac{\sigma_0^2}{p}w_0 \tag{4.25}$$

where n is the number of samples. For a budget p this is given by $n = \frac{p}{w_0}$. With the terms for both of these errors, we obtain the condition under which the MSE of a MFMC estimator is lower than the MSE of a standard MC estimator. It holds that $MSE[\hat{E}] < MSE[\hat{E}_{MC}]$ if and if only

$$\left(\sum_{i=0}^{k}\sqrt{w_i(\rho_{0,i}-\rho_{0,i+1}))}\right)^2 < w_0 \tag{4.26}$$

which is equivalent to

$$\left(\sum_{i=0}^{k}\sqrt{\frac{w_i}{w_0}(\rho_{0,i}-\rho_{0,i+1}))}\right)^2 < 1. \tag{4.27}$$

Furthermore, the smaller the value of the term on the left side is, the higher the variance reduction and the smaller the $MSE[\hat{E}]$ will be. We call this term the *variance reduction ratio* γ with

$$\gamma = \frac{MSE[\hat{E}]}{MSE[\hat{E}_{MC}]} = \left(\sum_{i=0}^{k}\sqrt{\frac{w_i}{w_0}(\rho_{0,i}-\rho_{0,i+1}))}\right)^2. \tag{4.28}$$

From this ratio, we can derive how to best choose the surrogate models to obtain a high variance reduction and therefore a low MSE. Both the costs of a model and its correlation coefficient contribute to the value of γ. We examine the characteristics a model should have to best decrease the variance reduction ratio. Firstly, γ is directly proportional to the costs w_i of the low-fidelity model $f^{(i)}$. This means that the less expensive a model is to evaluate, the higher its contribution to the variance reduction is because it leads to a smaller γ. Secondly, in the single terms of the sum in γ, the correlation coefficients $\rho_{0,i}$ always appear in conjunction with the correlation coefficient

of the following model $f^{(i+1)}$, i.e. in the difference $(\rho_{0,i} - \rho_{0,i+1})$. Therefore, γ will be low if that difference is low which is the case if the squared correlation coefficients $\rho_{0,i}^2$ and $\rho_{0,i+1}^2$ are of similar value. We want to emphasize this point: to achieve a high variance reduction, it is not only important for the costs of the low-fidelity models to be small, but the interaction of the different models plays a role as well, not just their individual correlation coefficients. Therefore, it is important to choose the models with regards to one another, not just according to their relationship to the high-fidelity model alone. Subsection 4.4.2 looks further at the problem of choosing the low-fidelity models in such a way that the MSE of an estimator based on these models is low.

4.4.2. Model selection

We discussed in Subsection 4.4.1 how to choose the control variate coefficients and the number of model evaluations for a given set of models that fulfill (4.19) and (4.20). With regards to the variance reduction ratio (4.28), we have seen that in order for the MFMC estimator to exhibit a low MSE, the models should be chosen such that their interaction leads to a low value of γ. This subsection presents an algorithm that, given an arbitrary set of a high-fidelity and k low-fidelity models, chooses a subset of these models which can be ordered as in (4.19) and fulfills (4.20). Additionally, these models will be selected such that out of all possible subsets, the chosen set leads to the MFMC estimator with the lowest MSE/variance.

This model selection algorithm was first proposed in [25]. We present it in Algorithm 3. As input, the algorithm takes the available models $f^{(0)}, f^{(1)}, ..., f^{(k)}$, including the high-fidelity model, the standard deviation σ_0 of the high-fidelity model and the correlation coefficients $\rho_{0,0}, \rho_{0,1}, ..., \rho_{0,k}$ and costs $w_0, w_1, ..., w_k$ of the models. The goal of the algorithm is to compare all possible subsets of models $\mathcal{M}$ with each other and find the subset $\mathcal{M}^*$ which will cause the lowest variance v^* of the resulting MFMC estimator. This is done by computing the variance v for each of the subsets according to (4.24) and selecting the set where this value is the lowest.

The algorithm first ensures the condition (4.19) by reordering the models if necessary. Because the variance depends on the computational budget p, we need to choose a value for it. The algorithm sets it to the runtime of the high-fidelity model $p = w_0$, however since this is never changed during the execution and the dependence of the estimator variance on p is linear, p could also be chosen differently and the result of the algorithm would be the same. The set $\mathcal{M}^*$ which will at termination of the algorithm contain the optimal subset of models is initialized with the high-fidelity model, ensuring that the returned set is not empty. We know the resulting variance of only using a high-fidelity model, which is equivalent to standard Monte Carlo sampling for a budget p, to be $\frac{\sigma_0}{p} w_0$ (see (4.25)). We initialize v^*, which will at any point in the

Algorithm 3 Model selection

Require: models $f^{(0)}, f^{(1)}, ..., f^{(k)}, \sigma_0, \rho_{0,0}, \rho_{0,1}, ..., \rho_{0,k}, w_0, w_1, ..., w_k$
  Reorder the models if $|\rho_{0,0}| > |\rho_{0,1}| > ... > |\rho_{0,k}|$ (cond. 4.19) is not fulfilled
  Set the computational budget $p = w_0$
  Initialize the set of models $\mathcal{M}^* = \{f^{(0)}\}$ and set $v^* = \frac{\sigma_0^2}{p} w_0$.
  for $\mathcal{M} \subseteq \{f^{(0)}, f^{(1)}, ..., f^{(k)}\}$ **do**
    if $f^{(0)} \notin \mathcal{M}$ **then**
      continue
    end if
    Set $l = |\mathcal{M}|$ to the number of models in $\mathcal{M}$
    Assign new indices $i_1, ..., i_l$ to the models in $\mathcal{M}$ s.t. $|\rho_{0,i_1}| > ... > |\rho_{0,i_l}|$
    Set $\rho_{0,i_{k+1}} = 0$
    Check that for $j = 1, ..., k$ it holds

$$\frac{w_{i_{j-1}}}{w_{i_j}} > \frac{\rho_{0,i_{j-1}}^2 - \rho_{0,i_j}^2}{\rho_{0,i_j}^2 - \rho_{0,i_{j+1}}^2} \quad \text{(cond. (4.20))} \tag{4.29}$$

    if cond. (4.20) does not hold for all $j = 1, ..., k$ **then**
      continue
    end if
    Compute

$$v = \frac{\sigma_0^2}{p} \left(\sum_{j=1}^{l} \sqrt{w_{i_j} (\rho_{0,i_j}^2 - \rho_{0,i_{j+1}}^2)} \right)^2 \tag{4.30}$$

    if $v < v^*$ **then**
      $\mathcal{M}^* = \mathcal{M}$
      $v^* = v$
    end if
  end for
  return $\mathcal{M}^*$

execution of the algorithm hold the lowest variance encountered so far by the set of models currently in $\mathcal{M}^*$, to this value.

With this, all of the subsets of the set of available models can be checked for their variance. The algorithm chooses a subset $\mathcal{M}$ and first ensures that the high-fidelity model $f^{(0)}$ is part of it and that the selection satisfies condition (4.20). If not, this set is discarded. Then the variance v of an MFMC estimator that uses all of the models in

$\mathcal{M}$ is computed according to (4.24). If this is the lowest variance encountered so far, we store the set $\mathcal{M}$ in $\mathcal{M}^*$ and the corresponding variance in v^*. After checking all of the possible subsets of $\{f^{(0)}, f^{(1)}, ..., f^{(k)}\}$, the algorithm terminates with the subset that will result in the lowest estimator variance and therefore, the lowest MSE. This is the set of models that, out of the available models, should be used to construct a MFMC estimator that is as accurate as possible.

The authors of [25] also note that since the algorithm iterates over all subsets of $\{f^{(0)}, f^{(1)}, ..., f^{(k)}\}$, its complexity is in $\mathcal{O}(2^k)$. However, in practice usually only few models are available, i.e. $k \leq 10$, and therefore the exponential runtime does not pose a problem. We now have a means to decide which of the available models to use in computing an MFMC estimator. The following section summarizes how to obtain such an estimator.

4.5. Multifidelity Monte Carlo Sampling

In the previous sections of this chapter, we have described all the theory necessary to construct a multifidelity estimator with the goal of achieving a lower MSE than that of a standard Monte Carlo estimator. Since it is based on standard Monte Carlo sampling, we call this technique multifidelity Monte Carlo ampling. We sum up this approach in Subsection 4.5.1 and address the finding of unknown correlation coefficients, variances and runtimes in Subsection 4.5.2.

4.5.1. MFMC algorithm

Algorithm 4 summarizes the procedure of multifidelity Monte Carlo sampling. As inputs, the algorithm takes the high-fidelity and the low-fidelity models $f^{(0)}, f^{(1)}, ..., f^{(k)}$, their variances $\sigma_0^2, \sigma_1^2, ..., \sigma_k^2$, computational costs $w_0, w_1, ..., w_k$ and the correlation coefficients $\rho_{0,0}, \rho_{0,1}, ..., \rho_{0,k}$ of the respective models with the high-fidelity model.

In Algorithm 3, we have described how an optimal set of models out of all the available models to minimize the MSE of the resulting estimator. With the help of this algorithm, the multifidelity Monte Carlo sampling algorithm first ensures that the input models have been chosen in accordance to the model selection algorithm. If they have not, an estimator with a lower MSE can be achieved. It then computes the control variate coefficients $\alpha_1, ..., \alpha_k$ and the number of model evaluations $m_0, m_1, ..., m_k$ as described in Subsection 4.4.1. Next, m_k realizations of the input $\mathbf{X}$ of the models are drawn and the models $f^{(i)}$ are evaluated for the first m_i samples. From these evaluations, the MFMC estimators for mean and variance are computed and returned.

Algorithm 4 Multifidelity Monte Carlo Sampling

Require: $f^{(0)}, f^{(1)}, ..., f^{(k)}, \sigma_0, \sigma_1, ..., \sigma_k, \rho_{0,0}, \rho_{0,1}, ..., \rho_{0,k}, w_0, w_1, ..., w_k, p$
if $f^{(0)}, f^{(1)}, ..., f^{(k)}$ are not the result of Algorithm 3 **then**
Restart the algorithm for a set of models chosen by Algorithm 3
end if
Set the control variate coefficients $\alpha_1, ..., \alpha_k$ to

$$\alpha_i = \frac{\rho_{0,i}\sigma_0}{\sigma_i} \quad \text{for } i = 1, ..., k \tag{4.31}$$

Set $\mathbf{r} = (r_0, r_1, ..., r_k)^T$ to

$$r_i = \sqrt{\frac{w_0(\rho_{0,i}^2 - \rho_{0,i+1}^2)}{w_i(1 - \rho_{0,1})}} \quad \text{for } i = 1, ..., k \tag{4.32}$$

Set the number of model evaluations $m_0, m_1, ..., m_k$ to

$$m_0 = \frac{p}{\mathbf{w}^T\mathbf{r}} \quad , \quad m_i = r_i m_0 \quad \text{for } i = 1, ..., k \tag{4.33}$$

Draw realizations $\mathbf{x_1}, ..., \mathbf{x_{m_k}}$ of the input vector X
for $0 \leq i \leq k$ **do**
Evaluate model $f^{(i)}$ for samples $\mathbf{x_1}, ..., \mathbf{x_{m_i}}$
end for
Compute MFMC estimators $\hat{E}$ and $\hat{V}$ for mean and variance of the high-fidelity output as (see Subsection 4.3.1 for details)

$$\hat{E} = \bar{E}_{m_0}^{(0)} + \sum_{i=1}^{k} \alpha_i(\bar{E}_{m_i}^{(i)} - \bar{E}_{m_{i-1}}^{(i)}) \quad \text{and} \quad \hat{V} = \bar{V}_{m_0}^{(0)} + \sum_{i=1}^{k} \alpha_i(\bar{V}_{m_i}^{(i)} - \bar{V}_{m_{i-1}}^{(i)}) \tag{4.34}$$

return $\hat{E}$, $\hat{V}$

4.5.2. Preprocessing

We note that Algorithm 4 requires the variances of the models, their costs and correlation coefficients. These quantities are typically not known. In practice we therefore need to estimate them beforehand. This can be done by drawing a number n of samples from the input domain, evaluating each of the models $f^{(0)}, f^{(1)}, ..., f^{(k)}$ for these samples and using the sample variance and sample correlation coefficient estimator to compute the unknown quantities. While evaluating the models, the time it takes to get the results can be measured and used to obtain estimates for the runtimes of the models. We summarize this preprocessing procedure as we will employ it later in Algorithm 5.

Algorithm 5 Preprocessing

Require: models $f^{(0)}, f^{(1)}, ..., f^{(k)}$, number of samples n
  Draw realizations $\mathbf{x_1}, ..., \mathbf{x_n}$ of the input vector $\mathbf{X}$
  for $0 \leq i \leq k$ **do**
    Evaluate model $f^{(i)}$ for samples $\mathbf{x_1}, ..., \mathbf{x_n}$ and store the time w_j it takes for evaluation j in $\mathbf{w_i} = (w_1, ..., w_n)^T$
  end for
  for $0 \leq i \leq k$ **do**
    Compute the sample variance $\bar{\sigma}_i^2$ of model $f^{(i)}$ according to (3.7)
    Compute the sample correlation coefficient $\bar{\rho}_{0,i}$ according to (3.8)
    Average the model evaluation costs from $\mathbf{w_i}$ to obtain the runtime $\bar{w}_i$ of model $f^{(i)}$:

$$\bar{w}_i = \frac{1}{n} \sum_{i=1}^{n} w_i \tag{4.35}$$

  end for
  return $\bar{\sigma}_0^2, \bar{\sigma}_1^2, ..., \bar{\sigma}_k^2, \bar{\rho}_{0,0}, \bar{\rho}_{0,1}, ..., \bar{\rho}_{0,k}, \bar{w}_0, \bar{w}_1, \bar{w}_k$

The work [25] shows experimentally that in practice it usually is sufficient to draw samples for a low number n because the effect small changes in these estimates have on the number of evaluations and the control variate coefficients is small. The work also notes that while some costs can follow from computing these estimates beforehand, often results are available from previous simulations that can be used instead. For our experiments in which we show results for different computational budgets and different model selections, these numbers also need to be computed only once for each model. Before we show these results, we present the process of multifidelity Monte Carlo sampling on a small numerical example.

4.6. Illustrative Example

In this section, we use an analytical function to demonstrate the MFMC algorithm. We further examine the MSE of the obtained estimators and show the improvement in precision compared to standard Monte Carlo estimators.

For this purpose we use the *Borehole function* as a high-fidelity model $f^{(0)}$. This function is a popular benchmark used in testing UQ techniques. It models the scenario of drilling a borehole from the ground through two aquifers. The value of the function gives the water flow rate in $\frac{m^2}{yr}$ through the borehole. For more details, see [20, 36]. The

function $f^{(0)} : \mathbb{R}^8 \to \mathbb{R}$ is given by

$$f^{(0)} = \frac{2\pi T_u (H_u - H_l)}{\ln(r/r_w)\left(1 + \frac{2LT_u}{\ln(r/r_w) r_w^2 K_w} + \frac{T_u}{T_l}\right)} \tag{4.36}$$

This function depends on eight parameters that are subject to uncertainty. We summarize them and their distributions in Table 4.1. We employ multifidelity Monte Carlo sampling to estimate mean and variance of the function output and compare the effectiveness of these estimators with that of standard Monte Carlo estimators. For the MFMC analysis, we use one low-fidelity model $f^{(1)}$. This is derived from the high-fidelity model as

$$f^{(1)} = \frac{5 T_u (H_u - H_l)}{\ln(r/r_w)\left(1.5 + \frac{2LT_u}{\ln(r/r_w) r_w^2 K_w} + \frac{T_u}{T_l}\right)} \tag{4.37}$$

and was proposed in [42]. Before we can run the MFMC algorithm, we need to find estimates for the models' correlation coefficients and variances. We obtain them as described in Algorithm 5. Because both high- and low-fidelity model are simple analytical functions that are very fast to evaluate and in reality have very similar evaluation times, we prescribe their runtimes by $w_0 = 10^{-3}s$ and $w_1 = 10^{-4}s$. To estimate the correlation coefficient and variances, we use $n = 10^7$ samples. The results are summarized in Table 4.2.

Before we give the practical results we obtained from executing the algorithm, we examine the variance reduction ratio γ (4.28) of the MFMC algorithm for this case. Recall that this ratio describes the reduction in the variance of the MFMC mean estimator $\hat{E}_{MFMC}$ compared to the standard Monte Carlo estimator $\hat{E}_{MC}$. It holds that $MSE[\hat{E}_{MFMC}] = \gamma MSE[\hat{E}_{MC}]$. Computing this ratio directly from the estimates we obtained for the correlation coefficient and the runtimes we prescribed, we obtain for this case

$$MSE[\hat{E}_{MFMC}] \approx 0.10 \cdot MSE[\hat{E}_{MC}]. \tag{4.38}$$

This means that for these models, we expect the precision of the MFMC mean estimator to increase by roughly one order of magnitude compared to the standard MC estimator. Using (4.24) and (4.25), we show this relationship of the expected MSEs in Figure 4.3. We refer back to this result when we describe our practical findings.

We now run multifidelity Monte Carlo sampling as shown in Algorithm 4. We obtain estimates for different computational budgets $p \in \{10^2, 5 \cdot 10^2, 10^3, 5 \cdot 10^3, 10^4\}$. These budgets are given in seconds. Note that the coefficient α_1 is independent of p (see (4.21)) and in this example is computed as 1.25664286124. The optimal numbers of model evaluations are given in Table 4.3. Observe that m_0 and m_1 are directly proportional to

parameter	**probability distribution**
radius of the borehole r_w	$\mathcal{N}\ (0.10, 0.0161812)$
radius of influence r	$\mathcal{LN}\ (7.71, 1.0056)$
transmissivity of the upper aquifer T_u	$\mathcal{U}(63070, 115600)$
potentiometric head of the upper aquifer H_u	$\mathcal{U}(990, 1110)$
transmissivity of the lower aquifer T_l	$\mathcal{U}(63.1, 116)$
potentiometric head of the lower aquifer H_l	$\mathcal{U}(700, 820)$
length of the borehole L	$\mathcal{U}(1120, 1680)$
hydraulic conductivity of the borehole K_w	$\mathcal{U}(9855, 12045)$

Table 4.1.: The eight input parameters used in the Borehole function. $\mathcal{N}(\mu, \sigma^2)$ denotes a normal (Gaussian) distribution and $\mathcal{LN}(\mu, \sigma^2)$ denotes a log-normal distribution with mean μ and variance σ^2.

	costs w_i $[s]$	standard deviation σ_i
$f^{(0)}$	10^{-3}	28.839468998
$f^{(1)}$	10^{-4}	22.9496143156

Table 4.2.: The costs and standard deviations of the high- and low-fidelity model employed in the numerical example of the Borehole function. The correlation coefficient of the two models is estimated as $\rho_{0,1} = 0.999999999998$.

the time budget and grow linearly with p (see (4.23)). The reason that these numbers do not scale exactly with p in this example is because the number of model evaluations are rounded from rational numbers to integers (see Section 4.4). We further compute standard Monte Carlo estimators using only the high-fidelity model $f^{(0)}$ as described in Algorithm 1.

In order to be able to judge the accuracy of the resulting MC and MFMC estimators, we compute reference estimators for the Borehole function. For this, we use standard Monte Carlo sampling with 10^8 i.i.d. samples and obtain $\mu = \mathbb{E}[f^{(0)}] \approx 74.0913128804$ and $\sigma^2 = \mathrm{Var}[f^{(0)}] \approx 831.301796299$.

Because the MFMC algorithm is optimized to minimize the MSE of the mean estimator, we compute the MSE with respect to the mean estimator $\hat{E}$. In order to achieve smoother plots, we compute the MSE as an average of 20 runs of the standard MC and MFMC algorithm which each yield estimators $\hat{E}_i$, $i = 1, \ldots, 10$. The MSE is obtained as follows:

$$MSE = \frac{1}{20} \sum_{i=1}^{10} (\mu - \hat{E}_i) \tag{4.39}$$

p	m_0	m_1
100	1	1420696
500	4	5682781
1000	8	11365562
5000	36	51145027
10000	71	100869358

Table 4.3.: The optimal number of model evaluations for different time budgets p for the example of the Borehole function problem.

The achieved mean squared errors of the estimators using standard MC and MFMC for the mean of the Borehole function are reported in Figure 4.4. Note the logarithmic scale on both axes. As shown in (4.24) and (4.25), it is clearly visible that for both standard MC and MFMC an increase of the computational budget p leads to a direct decrease in the MSE of the estimators. Using the MFMC algorithm with the second model $f^{(1)}$ in addition to $f^{(0)}$ reduces the MSE of the estimator by about one order of magnitude for a given budget. This confirms the theoretical estimate of the mean squared error we gave in (4.38).

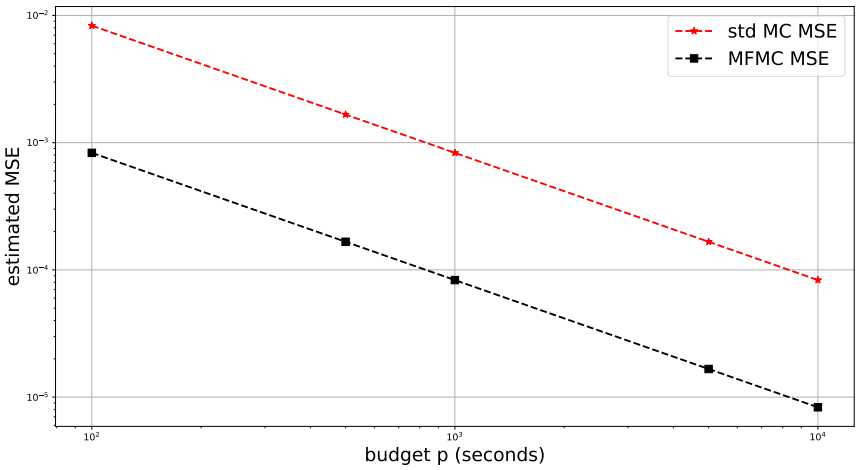


Figure 4.3.: The expected decay of the MSE of the standard Monte Carlo estimator and MFMC estimator for the mean of the Borehole function $f^{(0)}$.

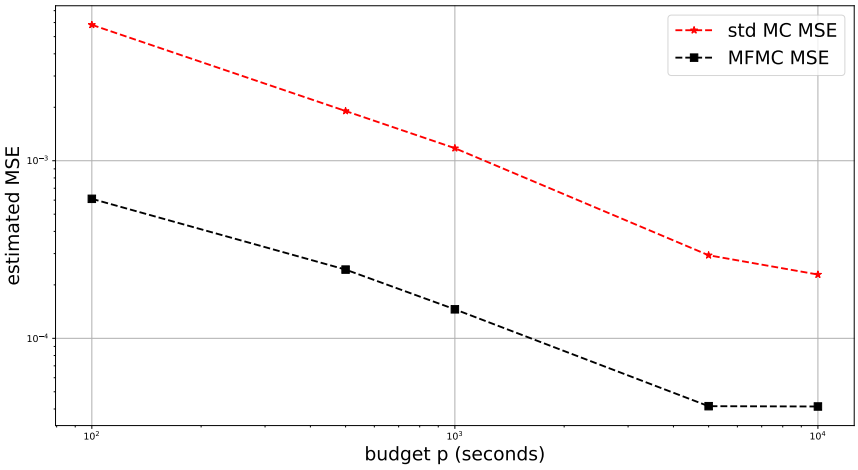


Figure 4.4.: The MSE of the standard Monte Carlo estimator and MFMC estimator for the mean of the Borehole function $f^{(0)}$. The MFMC estimator uses the low-fidelity model $f^{(1)}$ in addition to the high-fidelity model.

5. Results

This chapter presents our results from performing uncertainty propagation with the MFMC algorithm on two test cases of plasma microturbulence analysis. We examine the so-called *Cyclone Base Case* (CBC) where we first study a version with three uncertain input parameters (Section 5.1) to gain some first insights into the results that can be achieved with MFMC on this type of problem. We then move on to an extended CBC scenario with eight uncertain inputs to see the applicability of MFMC to a higher-dimensional and more challenging test case (Section 5.2). For both cases, we compute MFMC estimators of mean and variance of the GENE output and compare their accuracy to that of standard MC estimators.

5.1. Modified Cyclone Base Case (3D)

This section examines a modified version of the CBC test case with three stochastic parameters. Subsection 5.1.1 summarizes the physics of this scenario and describes the uncertainties in the input parameters. In Subsection 5.1.2, the construction of the low-fidelity models used for MFMC in this test case is described. Subsections 5.1.3 and 5.1.4 show the results we obtained from applying the MFMC algorithm to this scenario.

5.1.1. Description of the test case

This test case is a popular benchmark in gyrokinetics and represents a simplified description of turbulent transport as occurring in the ITER reactor. The type of here is a linear local, i.e. flux-tube, simulation for which GENE's linear eigenvalue solver is employed as a high-fidelity model.

The original version of this test case [6] models electrons and a single species of ions such that $n_i = n_e$ and $T_i = T_e$, i.e. the densities n_i and n_e of ions and electrons are assumed to be equal, as are their temperatures T_i and T_e. In order to create a more realistic scenario, the version of this test case used here [8] models electrons fully gyrokinetically. This means that the electrons' logarithmic temperature gradient $-L_s\delta_x \ln T_e$ is not assumed to be equal to the ions' temperature gradient $-L_s\delta_x \ln T_i$, but has to be taken into account as well. Plasmas are subject to quasi-neutrality, which means that the number of (singly charged) ions and electrons in a plasma are equal [41].

parameter	probability distribution
ion/electron log density gradient $-L_s\delta_x \ln n$	$\mathcal{U}(1.665, 2.775)$
ion log temperature gradient $-L_s\delta_x \ln T_i$	$\mathcal{U}(7.500, 12.500)$
electron log temperature gradient $-L_s\delta_x \ln T_e$	$\mathcal{U}(7.500, 12.500)$

Table 5.1.: The three uncertain input parameters for the 3D CBC test case and their probability distributions.

For this reason, the logarithmic density gradients $-L_s\delta_x \ln n_i$ and $-L_s\delta_x \ln n_e$ of the ions and electrons are modelled as equal. These three different quantities are the uncertain input parameters for this test case. We model them as independent uniform random variables. Their input ranges are summarized in Table 5.1. Further input parameters which for this scenario are fixed at deterministic values are modelled as stochastic inputs in the eight-dimensional version of this test case.

We are interested in one output obtained from this simulation. That output is the growth rate $\gamma[c_s/L_s]$ of the dominant eigenmode. Here, $c_s = \sqrt{T_e/m_i}$ is the ion sound speed (where m_i is the ion mass) and L_s is the characteristic length [8]. The goal of the uncertainty analysis is to firstly estimate the mean of this output based on the three uncertain input parameters. In addition to this, we estimate the standard deviation of the output. This gives a measure for how reliable the mean prediction is, i.e. if the standard deviation is low, then the true mean is very likely to be close to the value of the estimator, while for a high standard deviation this is less probable.

We perform simulations for normalized perpendicular wavenumbers $k_y\rho_s = 0.3, 0.8$. These values control the size of the local simulation box. For the CBC test case, it is known that at around $k_y\rho_s = 0.6$, there is a mode transition from an ITG mode to a trapped-electron-mode (TEM)/ETG hybrid mode [8]. For this reason, we choose the $k_y\rho_s$ for our runs such that both modes can be examined. Simulations for different $k_y\rho_s$ are independent of each other and can therefore be performed in parallel. For the application of MFMC to this scenario, this also means that low-fidelity models need to be constructed for both of the examined wavenumbers independently of each other. We present the employed models in the following.

5.1.2. Low-fidelity models

In order to create low-fidelity models to be used with MFMC, we apply the methods presented in Sections 2.3 and 2.4. We construct for both of the respective $k_y\rho_s$ a sparse grid (SG) approximation $f^{(1)}$ of the high-fidelity model $f^{(0)}$ and a machine-learning (ML) model $f^{(2)}$ based on an artificial neural network .

The accuracy and computational costs of the SG surrogate can be controlled by the

$k_y\rho_s$	SG corr. coefficient $\rho_{0,1}$	ML corr. coefficient $\rho_{0,2}$
0.3	0.99897	0.99987
0.8	0.98192	0.97085

Table 5.2.: The estimated correlation coefficient of the low-fidelity models with the high-fidelity model.

$k_y\rho_s$	GENE std. dev. σ_0	SG std. dev. σ_1	Ml std. dev. σ_2
0.3	0.13049	0.12015	0.12861
0.8	0.27471	0.27348	0.26519

Table 5.3.: The estimated standard deviations of the three models employed in the 3D CBC test case.

values of the tolerances $\boldsymbol{\tau} = (tol_1, \ldots, tol_d, tol_{d+1})$. Recall that the first $d = 3$ tolerances give a threshold for the contribution of the three input directions to the model output while tol_{d+1} controls the contribution by the interaction of all the input parameters. The adaptive algorithm of [8] stops refinement of the grid in one direction if the contribution of this dimension falls lower than the specified tolerance. Therefore, setting a higher tolerance leads the algorithm to stop refinement earlier, while prescribing low tolerances makes the algorithm explore the directions further even if their contributions are low. This leads to more grid points on which an interpolation of GENE is constructed and with that, to computationally more expensive and accurate models. For $k_y\rho_s = 0.3$, we set $\boldsymbol{\tau} = (10^{-1}, 10^{-1}, 10^{-1}, 10^{-1})^T$, while for $k_y\rho_s = 0.8$, we use $\boldsymbol{\tau} = (10^{-6}, 10^{-6}, 10^{-6}, 10^{-3})^T$.

We construct machine-learning surrogates which perform regression via a neural network as a second type of low-fidelity model. By doing this, we exploit the fact that we have high numbers of previous simulations runs from GENE available which makes training a neural network simple. We use the set up of the neural network as described in Section 2.4 for both $k_y\rho_s$. For $k_y\rho_s = 0.3$, we train the model with $\approx 10^5$ input-output data sets, while for $k_y\rho_s = 0.8$ we only use $\approx 2,5 \cdot 10^4$ samples. This leads to the ML model for $k_y\rho_s = 0.3$ achieving a higher accuracy as the one for $k_y\rho_s = 0.8$. Furthermore, it is important to note for our later experiments that the ML model for $k_y\rho_s = 0.3$ has a higher correlation with the high-fidelity model than the SG model does.

The correlation coefficients for the SG and ML models were computed from 1000 model evaluations of each of the models. They are listed in Table 5.2. Standard deviations of the three models were computed in the same runs and are given in Table 5.3.

$k_y\rho_s$	GENE runtime $[s]$ w_0	SG runtime $[s]$ w_1	ML runtime $[s]$ w_2
0.3	260.1698	0.0009	0.0020
0.8	240.5124	0.0167	0.0017

Table 5.4.: The estimated runtimes of the three models employed in the 3D CBC test case.

GENE runs are performed on two Intel Xeon E5-2697 nodes from the CoolMUC-2 Linux cluster[1] at the Leibniz Supercomputing Centre. Evaluations of the low-fidelity models are obtained on an Intel Core i5-8250U CPU running at 1.60GHz. The model runtimes are measured on the latter according to Algorithm 5. GENE runtimes are obtained from averaging the runtimes of 1000 runs and low-fidelity runtimes are computed from 50000 model evaluations. They are summarized in Table 5.4.

In the following two subsections, we present our results from performing MFMC using combinations of GENE with these two low-fidelity models for both wavenumbers $k_y\rho_s$.

5.1.3. Results for $k_y\rho_s = 0.3$

We first give our results from using both standard MC (Algorithm 1) and MFMC (Algorithm 4) to estimate mean and variance of the high-fidelity model output for $k_y\rho_s = 0.3$. We use both techniques for computational budgets $p \in \{5 \cdot 10^2, 10^3, 5 \cdot 10^3, 10^4, 5 \cdot 10^4, 10^5\}$. These budgets are given in seconds. We first perform MFMC using only the SG low-fidelity model in conjunction with the high-fidelity model because we want to show how different selections of low-fidelity models can affect the estimators.

As we did for the example in Section 4.6, we first present the theoretical improvement that the MFMC mean estimator $\hat{E}_{MFMC,1}$ using one low-fidelity model is expected to achieve compared to a standard MC estimator. Substituting the estimated computational costs and standard deviations of the two models into (4.28) gives for this set up the following variance reduction:

$$\frac{MSE[\hat{E}_{MFMC,1}]}{MSE[\hat{E}_{MC}]} \approx 0.00222. \tag{5.1}$$

In our simple example from Chapter 4, we only achieved a variance reduction ratio of 0.10. In this more realistic test case of plasma microturbulence analysis, we can expect a reduction of the variance of the estimator, and therefore of its MSE, by about three orders of magnitude. This is due to the tremendous reduction in runtime compared to

[1] https://www.lrz.de/services/compute/linux-cluster/

the high-fidelity model which we achieved for the SG surrogate. The decay of the MSE is depicted in Figure 5.1 using (4.24) and (4.25). This plot represents the theoretical, ideal reduction of the MSE based on the assumption that all the estimated model parameters, i.e. variances, correlation coefficients and runtimes, are exact. Note that we examine the MSE of the MFMC mean estimator since the number of model evaluations and control variate coefficients are optimized with regards to the mean estimator.

As a reference for the true value of the model output's mean for $\gamma[c_s/L_s]$, we use a result obtained by performing standard MC with a SG surrogate. This SG model has been constructed by setting the tolerances of the adaptive sparse grid algorithm to $\boldsymbol{\tau} = (10^{-14}, 10^{-14}, 10^{-14}, 10^{-14})^T$ which yields a very accurate approximation of GENE. The lower runtime of the resulting model then allows using the MC algorithm with a high number of samples. In this case, 25000 model evaluations have been obtained to compute the reference estimator. This results in the estimate $\mu = \mathbb{E}[f^{(0)}] \approx 0.68091$. For the standard deviation, we obtain a reference of $\sigma \approx 0.13035$.

Furthermore, we perform 10 runs of the MC and MFMC algorithms for each of the budgets, obtaining estimates $\hat{E}_1, \ldots, \hat{E}_{10}$ for the mean value. We compute the MSE as an average over these quantities:

$$MSE = \frac{1}{10} \sum_{i=1}^{10} (\mu - \hat{E}_i). \tag{5.2}$$

Figure 5.2 shows the MSE of the computed MC and MFMC estimators of the mean of the model output $\gamma[c_s/L_s]$. The results confirm the preliminary estimates of the MSE decay obtained by examining the variance reduction of the MFMC estimator compared to the standard MC estimator. We can see that the MSE of the estimator directly decreases with increasing budgets (see (4.24)). Furthermore, by applying MFMC we can decrease the MSE of the standard MC estimator by about three orders of magnitude as we estimated by means of the variance reduction ratio.

In Algorithm 3, we presented an algorithm that, out of a given set of models, selects the subset of models that leads to a MFMC estimator with the lowest MSE that can be achieved with any of the models in the set. The algorithm does this by computing the variance of the resulting MFMC estimators and selecting the subset of models that leads to the lowest variance. Executing it for the three models we have for the current scenario, i.e. the high-fidelity model and the SG and ML low-fidelity models, this algorithm will return a set containing all three. This means that by using all three, we will achieve estimators with a lower MSE than the one we obtained using only the SG model as a low-fidelity model.

Computing the variance reduction ratio for MFMC using the two low-fidelity models

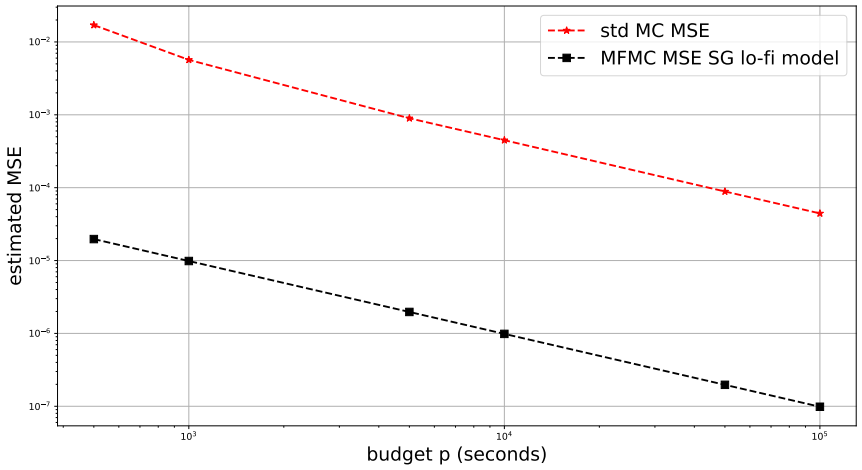


Figure 5.1.: The expected decay of the MSE of the standard Monte Carlo and MFMC estimator for the mean value of the growth rate $\gamma[c_s/L_s]$ for $k_y\rho_s = 0.3$. The MFMC estimator uses the high-fidelity model and the SG low-fidelity model.

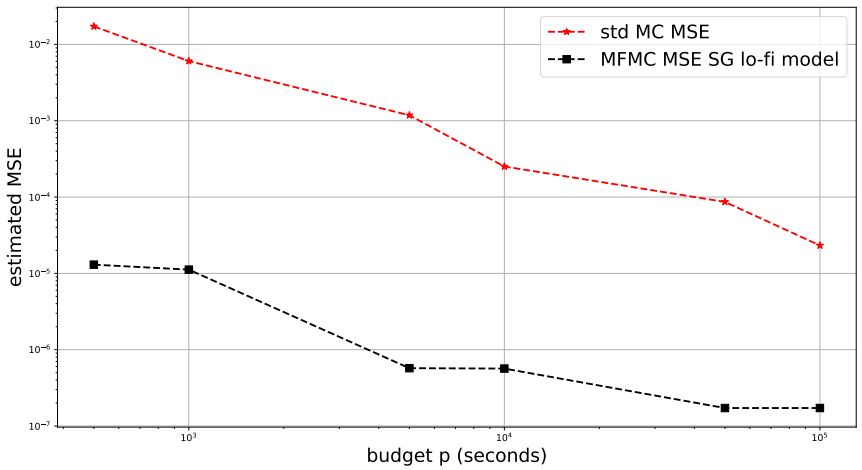


Figure 5.2.: The MSE of the standard Monte Carlo and MFMC estimator for the mean value of the growth rate $\gamma[c_s/L_s]$ for $k_y\rho_s = 0.3$ using the high-fidelity model and the SG low-fidelity model in the 3D CBC test case.

with regard to standard MC gives

$$\frac{MSE[\hat{E}_{MFMC,2}]}{MSE[\hat{E}_{MC}]} \approx 0.00032. \tag{5.3}$$

In the case of only using the SG model, we obtained a significant variance reduction ratio of $\gamma = 0.00222$ (5.1). This was due to the SG model being computationally less expensive than the high-fidelity model by a factor of the order of 10^5s and its correlation coefficient $\rho_{0,1}$ being very high. By adding the second low-fidelity model, we achieve even further variance reduction. We mentioned in Subsection 4.4.1 that in order for a set of models to reduce the MSE further than another combination, they need to compliment each other with respect to their correlation coefficients. This is the case here.

While the variance reduction from adding the second low-fidelity model to the combination of $f^{(0)}$ and $f^{(1)}$ is not as big as the change from standard MC to MFMC with one low-fidelity model, we can still obtain further variance reduction from the additional ML model. Figure 5.3 illustrates this. In Figure 5.4 we show our results from applying the MFMC algorithm using both low-fidelity models to the test case. Again, they confirm the theoretical results we obtained from computing the variance reduction. Results for $p = 10^5$ were not computed for this set of models.

Figure 5.5 depicts how MFMC distributes the model evaluations to the different models. For each combination of models we examined, the plot shows the fraction that m_0, m_1 and m_2 make up of the total number of evaluations over all models. Note the logarithmic scale on the y-axis. This distribution is valid for all time budgets p, because the number of model evaluations scales linearly with p (see (4.23)). In the case of MFMC with only the SG model as a low-fidelity model, only a very small number of evaluations falls to the high-fidelity model. Adding the ML model as a second low-fidelity model, most of the evaluations are still distributed to the SG model which, in this combination, is again the computationally cheapest model and the model with the lowest correlation coefficient. Here, before computing the optimal number of model evaluations and control variate coefficients, the models are reordered to satistfy (4.19).

Finally, Table 5.5 lists the MC and MFMC estimators computed for the model output's mean and variance for the different time budgets. Note that a budget of 500 only allows for one evaluation of the high-fidelity model (since $w_0 = 260.1698$) and that therefore, no estimate for the variance can be computed using standard MC for this budget. MFMC on the other hand uses the remaining budget for additional low-fidelity evaluations with which a variance estimator can be obtained.

For the purpose of understanding the model selection algorithm better, we obtained another ML surrogate with a correlation coefficient to high-fidelity model of $\rho \approx 0.77389$ and a runtime of $w \approx 0.00193$. While the runtime of this model is comparable to that of

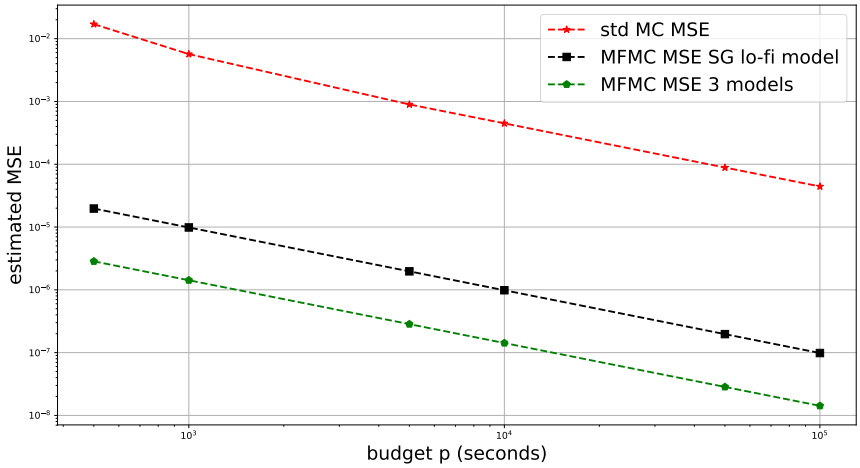


Figure 5.3.: The expected decay of the MSE of the standard Monte Carlo and MFMC estimators for the mean value of the growth rate $\gamma[c_s/L_s]$ for the 3D CBC test case with $k_y\rho_s = 0.3$. The first MFMC estimator uses only the SG model (black), the second uses the SG and the ML low-fidelity models (green).

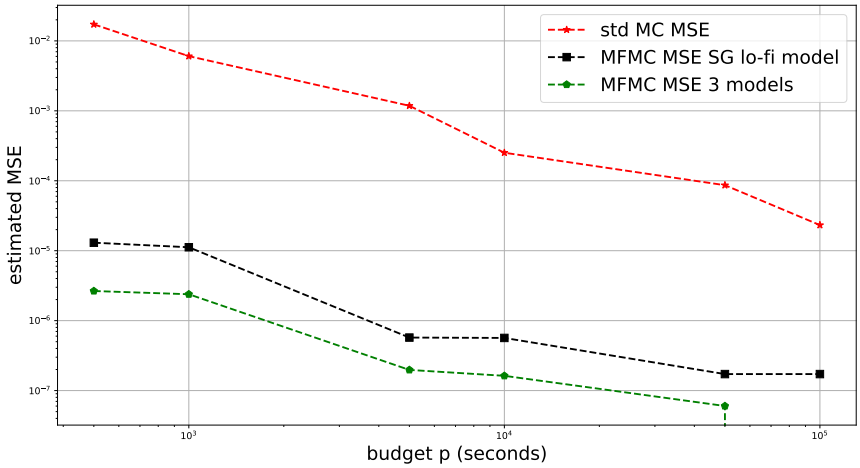


Figure 5.4.: The MSE of the standard Monte Carlo and MFMC estimators for the mean value of the growth rate $\gamma[c_s/L_s]$ for the 3D CBC test case with $k_y\rho_s = 0.8$. The first MFMC estimator uses only the SG model (black), the second uses the SG and the ML low-fidelity models (green).

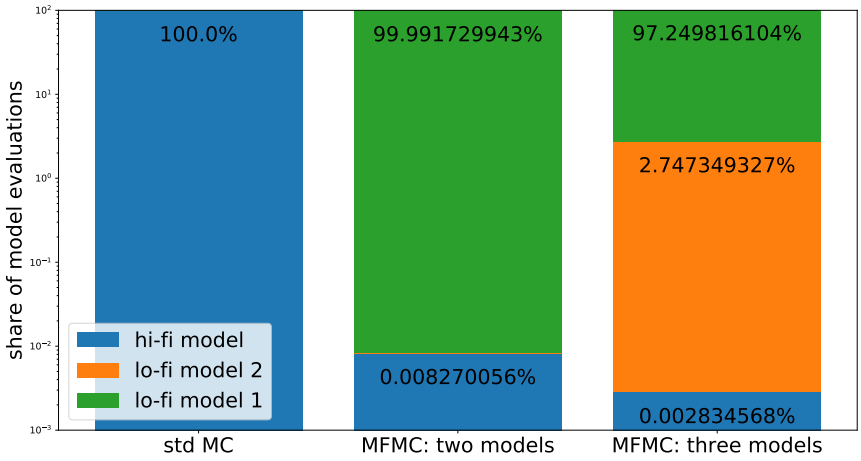


Figure 5.5.: The distribution of evaluations over the models for standard MC and MFMC with two and three models for $k_y\rho_s = 0.3$. On the logarithmic y-axis, the percentage of model evaluations that fall to the different models is given.

p	$\hat{E}_{MC}$	$\hat{E}_{MFMC,1}$	$\hat{E}_{MFMC,2}$	$\hat{\sigma}_{MC}$	$\hat{\sigma}_{MFMC,1}$	$\hat{\sigma}_{MFMC,2}$
500	0.72962867	0.68067796	0.6804246	-	0.12497945	0.12868827
1000	0.69809482	0.67982301	0.68106508	0.06459229	0.13115049	0.12914015
5000	0.6803166	0.68121959	0.68116393	0.11145522	0.12980779	0.12928115
10000	0.68458738	0.68122072	0.68101832	0.14912978	0.12985041	0.12948363
50000	0.68213214	0.68093813	0.6811262	0.13138754	0.12929114	0.12950445
100000	0.68100762	0.6812384	-	0.13243871	0.12964336	-

Table 5.5.: The estimated mean $\hat{E}$ and standard deviation $\hat{\sigma}$ of the model output $\gamma[c_s/L_s]$ for $k_y\rho_s = 0.3$ obtained with MC and MFMC estimators in the 3D CBC test case. The subscripts $MFMC,1$ and $MFMC,2$ refer to estimators obtained from MFMC using one and two low-fidelity models respectively.

the ML model used in the previous experiments, its correlation coefficient is significantly smaller. Running the model selection algorithm for the SG model and this ML model, the algorithm selects the high-fidelity and the SG model only. In this case, adding a second model actually increases the MSE of the MFMC estimators.

5.1.4. Results for $k_y\rho_s = 0.8$

This subsection shows our results for wavenumber $k_y\rho_s = 0.8$ in the three-dimensional CBC test scenario. As in the previous experiments, we first computed standard MC estimators for mean and variance of the high-fidelity model output $\gamma[c_s/L_s]$ for the same increasing budgets p. We then performed MFMC with first only the SG model created for this $k_y\rho_s$ and then added the ML model as a second low-fidelity model.

The runtime of the SG model we created for this scenario is higher than that of the model for $k_y\rho_s = 0.3$ and its correlation coefficient is lower. This is attributed to the different microinstability mode and more complex physics needed to model it. For this reason, constructing the SG approximation requires more grid points and yields a more expensive model. Both the runtime and correlation coefficient influence the variance reduction ratio (4.28), which in this case computes to

$$\frac{MSE[\hat{E}_{MFMC,1}]}{MSE[\hat{E}_{MC}]} \approx 0.039. \tag{5.4}$$

The plots in Figure 5.6 show the expected reduction of the MSE for MFCM with the SG model. To compute the MSE we achieved in the practical application of MFMC, we compute reference estimators for the mean $\mu \approx 0.55318$ and for the standard deviation $\sigma \approx 0.27666$. These results were obtained by performing standard MC using approximately 38000 samples of the high-fidelity model. With this, we estimate the MSE of the MC and MFMC estimators and obtain the results shown in Figure 5.7. These plots were again created by computing an average MSE over 10 runs of the MC and MFMC algorithms.

We then ran the model selection algorithm for all three models: the high-fidelity model, the SG low-fidelity model and the ML low-fidelity model. For this set of models, the algorithm again returns the selection of all three models, meaning that a MFMC estimator computed using all three will exhibit a lower MSE than an estimator using only one of the low-fidelity models. The variance reduction ratio for this set of models is given by

$$\frac{MSE[\hat{E}_{MFMC,2}]}{MSE[\hat{E}_{MC}]} \approx 0.037. \tag{5.5}$$

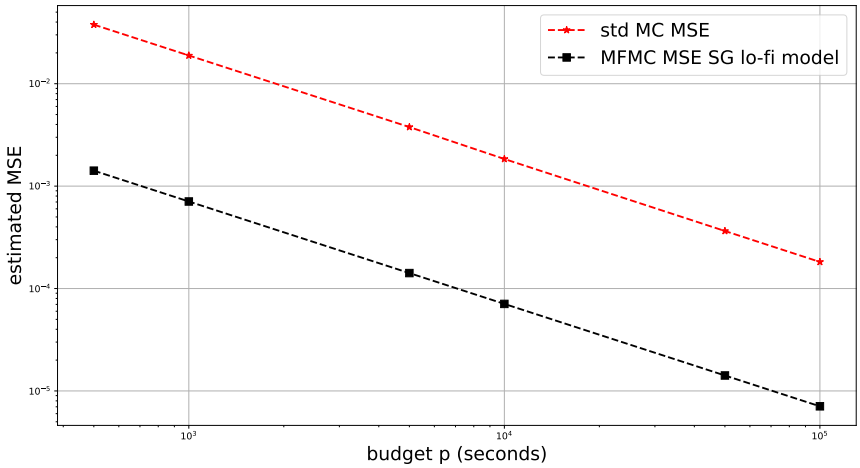


Figure 5.6.: The expected decay of the MSE of the standard Monte Carlo and MFMC estimator for the mean value of the growth rate $\gamma[c_s/L_s]$ for $k_y\rho_s = 0.8$ in the 3D CBC test case. The MFMC estimator uses the high-fidelity model and the SG low-fidelity model.

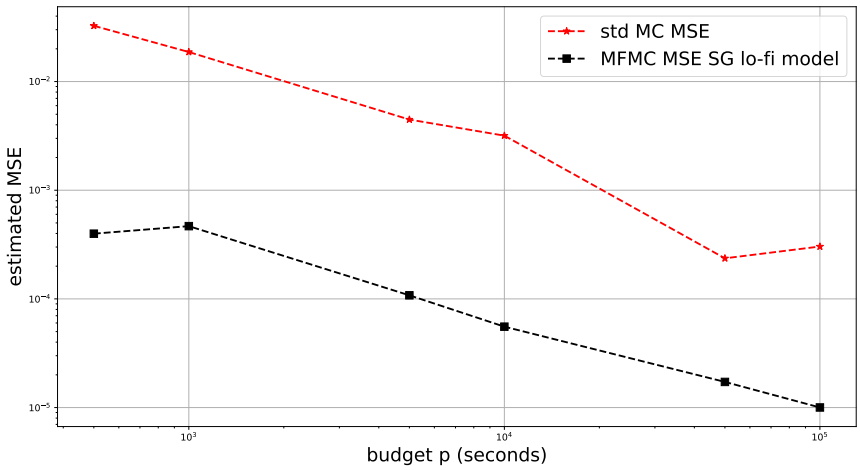


Figure 5.7.: The MSE of the standard Monte Carlo and MFMC estimator for the mean value of the growth rate $\gamma[c_s/L_s]$ for $k_y\rho_s = 0.8$ using the high-fidelity model and the SG low-fidelity model in the 3D CBC test case.

p	$\hat{E}_{MC}$	$\hat{E}_{MFMC,1}$	$\hat{E}_{MFMC,2}$	$\hat{\sigma}_{MC}$	$\hat{\sigma}_{MFMC,1}$	$\hat{\sigma}_{MFMC,2}$
500	0.54945094	0.5591043	0.56674721	0.41094231	0.27346354	0.2693183
1000	0.50797404	0.5605094	0.55732878	0.29885409	0.26297078	0.28351324
5000	0.56687098	0.55628552	0.55223438	0.28189931	0.27492931	0.28295497
10000	0.58152203	0.55665978	0.56177573	0.1938001	0.27186596	0.26959688
50000	0.5606377	0.55506742	0.55648918	0.27432992	0.27513511	0.27396548
100000	0.56078382	0.55578007	0.5560119	0.27523817	0.27242003	0.27514184

Table 5.6.: The estimated mean $\hat{E}$ and standard deviation $\hat{\sigma}$ of the model output $\gamma[c_s/L_s]$ for $k_y\rho_s = 0.8$ obtained with MC and MFMC estimators in the 3D CBC test case. The subscripts $MFMC,1$ and $MFMC,2$ refer to estimators obtained from MFMC using one and two low-fidelity models respectively.

Compared to the variance reduction factor 0.039 for only the SG model, this selection of models does not yield a large additional decrease of estimator's variance. This can also be seen in Figure 5.8.

Our results from computing the MSE of our obtained MFMC estimators with respect to the given reference mean estimator support this observation. The results from our simulations are depicted in Figure 5.9. In contrast to our results for $k_y\rho_s = 0.3$ shown in Figure 5.4, the MSE plots for $k_y\rho_s = 0.8$ still shows large fluctuations rather than a linear decay (on a logarithmic scale) as expected. For some of the computational budgets, the experimentally determined MSE even exceeds the MSE of the MFMC estimator that only uses the SG low-fidelity model. This can be attributed to the fact that for $k_y\rho_s = 0.8$, the runtime of the employed SG model is almost 20 times as high as the runtime of the SG model for $k_y\rho_s = 0.3$. Because of this, fewer model evaluations are possible in the given time frames. For comparison, for $p = 500$ this only allows for $m_0 = 2$ high-fidelity evaluations and $m_1 = 187$ (ML) and $m_2 = 3830$ (SG) low-fidelity evaluations, while for $k_y\rho_s = 0.3$ these numbers were $m_0 = 1$, $m_1 = 970$ (SG) and $m_2 = 34309$ (ML). A solution to achieve more accurate results could be to find a second low-fidelity model with a lower runtime but a comparable correlation coefficient, or increase the computational budget.

Figure 5.10 depicts the distribution of the model evaluations over the available models. Note that in this case, there is no reordering necessary when the second low-fidelity model is added because it holds that $|\rho_{0,1}| > |\rho_{0,2}|$. For this reason, the largest share of evaluations then falls to the newly added ML model, not the SG model as was the case for $k_y\rho_s = 0.3$. Table 5.6 lists the estimators obtained for $k_y\rho_s$ with both MC and MFMC using one and two low-fidelity models.

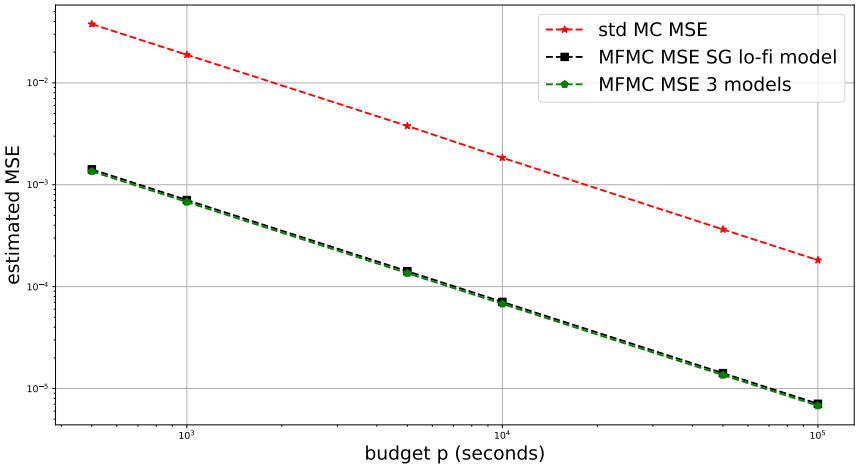


Figure 5.8.: The expected decay of the MSE of the standard Monte Carlo and MFMC estimators for the mean value of the growth rate $\gamma[c_s/L_s]$ for the 3D CBC test case with $k_y\rho_s = 0.8$. The first MFMC estimator uses only the SG model (black), the second uses the SG and the ML low-fidelity models (green).

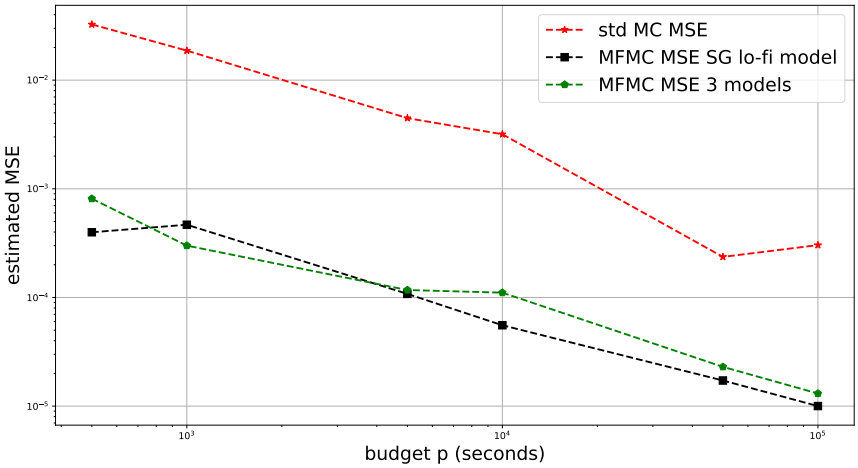


Figure 5.9.: The MSE of the standard Monte Carlo and MFMC estimators for the mean value of the growth rate $\gamma[c_s/L_s]$ for the 3D CBC test case with $k_y\rho_s = 0.8$. The first MFMC estimator uses only the SG model (black), the second uses the SG and the ML low-fidelity models (green).

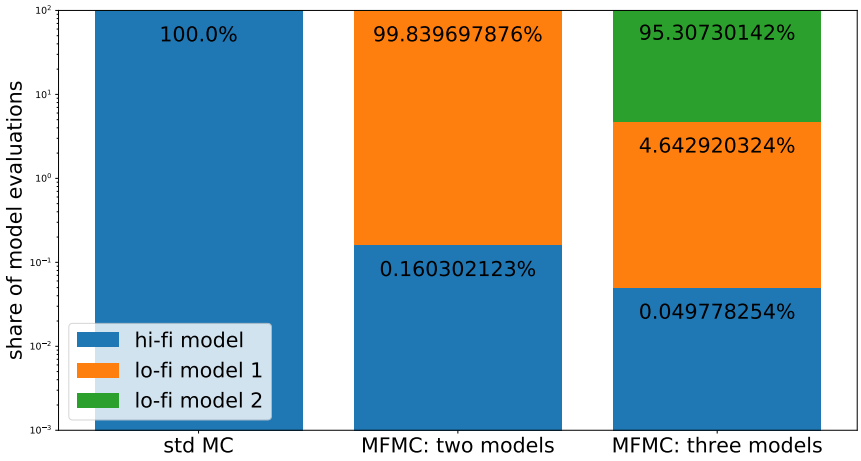


Figure 5.10.: The distribution of evaluations over the models for standard MC and MFMC with two and three models for $k_y\rho_s = 0.8$. On the logarithmic y-axis, the percentage of model evaluations that fall to the different models is given.

5.2. Modified Cyclone Base Case (8D)

This section summarizes our results from the application of the MFMC algorithm to the CBC test case that is modelled with eight uncertain input parameters. Subsection 5.2.1 explains the additions made to the three-dimensional version of this test case from the previous section to obtain this extended scenario. The low-fidelity model we use in the MFMC algorithm for this application is described in Subsection 5.2.2. Subsection 5.2.3 presents our results for this test case.

5.2.1. Description of the test case

This test case extends the version of the CBC that is given in Subsection 5.2.1. We expand the three-dimensional scenario to eight dimensions by modelling uncertainty in additional input parameters which have previously been set to a deterministic value. By doing this, we model a setting that is closer to realistic conditions inside a fusion reactor. The CBC test case with these eight uncertain inputs was first presented in [8].

The input parameters and their probability distributions for this case are listed in Table 5.7. The parameters are again modelled as independent. We keep the density and temperature gradients of the ions and electrons as we have modelled them before.

parameter	**probability distribution**
plasma beta β	$\mathcal{U}(0.598 \cdot 10^{-3}, 0.731 \cdot 10^{-3})$
collision frequency ν_c	$\mathcal{U}(0.238 \cdot 10^{-2}, 0.322 \cdot 10^{-2})$
ion/electron log density gradient $-L_s \delta_x \ln n$	$\mathcal{U}(1.665, 2.775)$
ion log temperature gradient $-L_s \delta_x \ln T_i$	$\mathcal{U}(7.500, 12.500)$
electron log temperature gradient $-L_s \delta_x \ln T_e$	$\mathcal{U}(7.500, 12.500)$
temperature ratio T_i/T_e	$\mathcal{U}(0.950, 1.050)$
magnetic shear $\hat{s}$	$\mathcal{U}(0.716, 0.875)$
safety factor q	$\mathcal{U}(1.330, 1470)$

Table 5.7.: The eight uncertain input parameters for the 8D CBC test case and their probability distributions.

Further uncertainty is introduced in the ratio T_i/T_e of the ion and electron temperatures themselves which we have previously assumed to be 1. Because we treat electrons fully gyrokinetically, electromagnetic effects in the plasma need to be considered. The stochastic parameter β describes their magnitude as a ratio of the kinetic to the magnetic pressure inside the plasma. To compute collisions between particles, we use the linearized Landau-Boltzmann collision operator as implemented in GENE. The frequency ν_c of collisions is now modelled as uncertain. The last two parameters we have previously modelled as deterministic characterise the magnetic field. We add uncertainties to the safety factor q which describes the relationship between the number of toroidal turns of a magnetic field line to the number of poloidal turns [8] and its derivative $\hat{s} = \frac{r}{q}\frac{\delta q}{\delta r}$ with respect to r, which is the radial coordinate labelling flux surfaces [8].

We again compute one model output, the growth rate $\gamma[c_s/L_s]$ of the dominant eigenmode. For this scenario, we examine the mean and variance of this output computed with MC and MFMC for $k_y\rho_s = 0.3$.

5.2.2. Low-fidelity model

For this scenario, we perform MFMC using one low-fidelity model. We again construct a SG surrogate for GENE using the approach from Section 2.3. For the tolerances used in the adaptive sparse grid algorithm, we prescribe $\boldsymbol{\tau} = (10^{-6}, \ldots, 10^{-6}, 10^{-6})^T$ and obtain a model $f^{(1)}$ with a correlation coefficient $\rho_{0,1} \approx 0.99999$. Runtimes for GENE and this SG model are measured on the machines as described above. They are given, along with the estimated standard deviations of both models, in Table 5.8.

	GENE $f^{(0)}$	SG model $f^{(1)}$
runtime $w_i[s]$	574.59450	0.10061
standard deviation σ_i	0.12999	0.13021

Table 5.8.: The runtimes and standard deviations of the high- and low-fidelity model for the 8D CBC test case for $k_y\rho_s = 0.3$.

5.2.3. Results for $k_y\rho_s = 0.3$

We now present our results from using standard MC and MFMC on the 8D CBC test case. As before, we estimate mean and variance of the model output with both techniques and examine the MSE of the obtained estimators. Because the runtime of the high-fidelity model for this test case exceeds a budget of 500, we apply the algorithm for the remaining budgets that have previously been used.

Computing the variance reduction from the data we have just given in Subsection 5.2.2, we obtain for this scenario

$$\frac{MSE[\hat{E}_{MFMC,1}]}{MSE[\hat{E}_{MC}]} \approx 0.0003. \tag{5.6}$$

This means that by performing MFMC with the SG model as a low-fidelity model, we can achieve about four orders of magnitude in the estimators' precision compared to standard MC for a fixed computational budget. The expected decay of the MSE is illustrated in Figure 5.11.

Our results from applying MC and MFMC are depicted in Figure 5.12. The MSE is computed with respect to a reference mean estimator obtained during the (deterministic) construction of the SG surrogate. We use $\mu \approx 0.67936$ and $\sigma \approx 0.13119$. We can see that again the estimated MSE is very close to our theoretical observations.

Figure 5.13 shows the distribution of model evaluations over the two models. We note that, as in the cases examined before, MFMC allocates most of the evaluations to the computationally cheaper SG model and only very few are given to the high-fidelity model (e.g. for $p = 10^5$, $m_0 = 40$ and $m_1 = 765683$).

Table 5.9 lists the estimates obtained for mean and variance of the high-fidelity output. Note that the budget $p = 1000$ only allows for one high-fidelity model evaluation and therefore, no variance can be computed using standard MC. Comparing these results with the estimators we computed in the three-dimensional CBC scenario (Table 5.5), we observe that the estimated values are very similar. This is due to the additional parameters that are modelled as uncertain here contributing little to the model output. In fact, [8] shows by performing sensitivity analysis on this test case that for $k_y\rho_s = 0.3$, the logarithmic electron temperature gradient $-L_s\delta_x \ln T_e$ has the largest impact on the

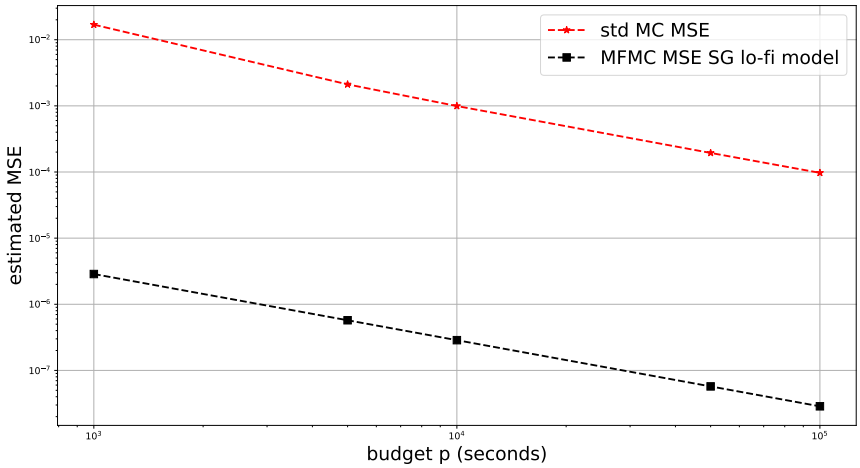


Figure 5.11.: The expected decay of the MSE of the standard Monte Carlo and MFMC estimator for the mean value of the growth rate $\gamma[c_s/L_s]$ for $k_y\rho_s = 0.3$ in the 8D CBC test case. The MFMC estimator uses the high-fidelity model and the SG low-fidelity model.

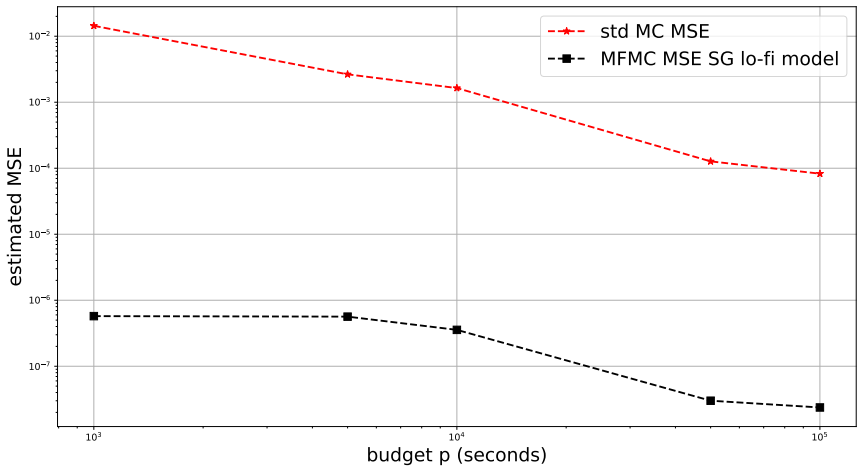


Figure 5.12.: The MSE of the standard Monte Carlo and MFMC estimators for the mean value of the growth rate $\gamma[c_s/L_s]$ for the 8D CBC test case with $k_y\rho_s = 0.3$ using the high-fidelity model and the SG low-fidelity model.

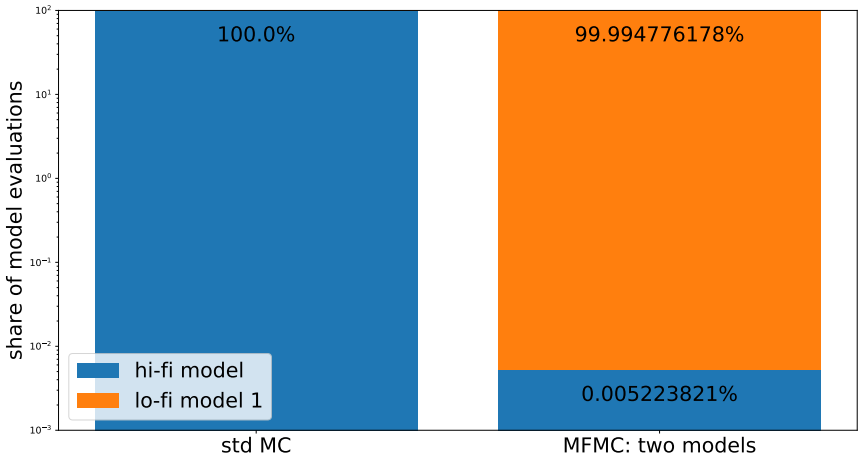


Figure 5.13.: The distribution of evaluations over the models for standard MC and MFMC with two models for $k_y\rho_s = 0.3$ in the 8D CBC test case. On the logarithmic y-axis, the percentage of model evaluations that fall to the different models is given.

p	$\hat{E}_{MC}$	$\hat{E}_{MFMC,1}$	$\hat{\sigma}_{MC}$	$\hat{\sigma}_{MFMC,1}$
1000	0.69485483	0.67953651	-	0.13059121
5000	0.6945134	0.67938304	0.15957415	0.13112354
10000	0.69078531	0.67938439	0.13393346	0.1309927
50000	0.67682187	0.67940607	0.12994909	0.13098227
100000	0.67802467	0.67946586	0.1341998	0.13102557

Table 5.9.: The estimated mean $\hat{E}$ and standard deviation $\hat{\sigma}$ of the model output $\gamma[c_s/L_s]$ for $k_y\rho_s = 0.3$ obtained with MC and MFMC estimators in the 8D CBC test case.

model output while the contributions from other parameters are negligible.

This simple experiment of adding one low-fidelity model to the high-fidelity model and computing MFMC estimators using both shows the usefulness of the MFMC algorithm. Investing the one-time effort in constructing the SG model for this test case means the runtimes of the estimation process can be decreased by a factor of

approximately 10^4 to obtain estimators of the same precision as before if MFMC instead of MC is used. Additional variance reduction could be achieved if further low-fidelity models are added.

6. Conclusion and outlook

The main focus of this thesis was the application of multifidelity Monte Carlo sampling in the quantification of uncertainty in a real-world application, plasma microturbulence analysis. We showed in detail how MFMC uses high- and low-fidelity models that describe the same physical phenomenon to reduce the variance and with that the MSE of standard MC estimators used in UQ. To demonstrate the benefits of MFMC, both of the algorithms were applied to the simulation of plasma microturbulence, a field to which UQ has only been introduced in recent years and which is of great interest in the study of nuclear fusion. To this end, simulations were conducted using the plasma physics code GENE as a gyrokinetic solver. Low-fidelity models that approximate GENE were constructed via interpolation on sparse grids and regression using an artificial neural network.

With these models, two test scenarios of plasma microturbulence analysis were examined. We applied the MC and MFMC algorithms to quantify uncertainty in a modified version of the popular Cyclone Base Case benchmark. In a first run, we studied a version of the CBC with three uncertain input parameters. For this test case, we performed simulations for two different wavenumbers $k_y\rho_s = 0.3, 0.8$. For both of them, we first executed MFMC using a SG model as a low-fidelity model and then added a ML model, both for increasing computational budgets. In both cases, MFMC with one low-fidelity model lead to a reduction of the MSE of a mean estimator for the model output of at least one order of magnitude. For $k_y\rho_s = 0.3$, the error could be reduced further by adding the ML model, while for $k_y\rho_s = 0.8$ the particular configuration of the SG and ML models chosen resulted in little improvement. We then applied both algorithms to a more realistic, eight-dimensional CBC scenario. With the little effort of only adding a very accurate SG surrogate to the high-fidelity model, MFMC achieved a large MSE reduction by about four orders of magnitude.

With these results, we demonstrated that MFMC can be a very effective tool in forward UQ by reducing the computational budget needed to generate mean and variance estimators of a given accuracy. We showed that adding multiple low-fidelity models can further improve this approach. By using models constructed via machine learning and interpolation, we illustrated how MFMC can be employed with low-fidelity models of any type and construction. Moreover, by the application of MFMC to test cases from plasma microturbulence analysis we found the MFMC algorithm to be

very promising to UQ studies in this field of research.

In the future, other low-fidelity models with varying runtimes and correlation coefficients could be added to the existing models to perhaps further improve the performance of MFMC on the test cases considered. Moreover, while the CBC test case employed here does model a real-world scenario, it is still a simplified description of the application. Therefore, MFMC should be applied to additional, more realistic test cases to fully study the benefits the algorithm can bring to UQ in plasma microturbulence analysis.

A. Appendix

A.1. Source code: Implementation of a machine-learning surrogate for Gene

```
def build_model():
    model = keras.Sequential([
        keras.layers.Dense(3, activation=tf.nn.relu, input_shape=(3,)),
        keras.layers.Dense(3, activation=tf.nn.relu),
        keras.layers.Dense(1)
    ])

    optimizer = tf.keras.optimizers.RMSprop(0.001)

    model.compile(loss='mean_squared_error',
                  optimizer=optimizer,
                  metrics=['mean_squared_error'])

    return model
```

Listing A.1: The Python function we use to build a machine-learning model.

List of Figures

List of Tables

Bibliography

[1] P. Benner, S. Gugercin, and K. Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Review*, 57(4):483–531, 2015.

[2] J.-P. Berrut and L. N. Trefethen. Barycentric lagrange interpolation. *SIAM review*, 46(3):501–517, 2004.

[3] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, Berlin, Heidelberg, 2006.

[4] A. Brizard and T. Hahm. Foundations of nonlinear gyrokinetic theory. *Reviews of Modern Physics*, 79, 2007.

[5] H.-J. Bungartz and M. Griebel. Sparse grids. *Acta numerica*, 13:147–269, 2004.

[6] A. M. Dimits, G. Bateman, M. A. Beer, B. I. Cohen, W. Dorland, G. W. Hammett, C. Kim, J. E. Kinsey, M. Kotschenreuther, A. H. Kritz, L. L. Lao, J. Mandrekas, W. M. Nevins, S. E. Parker, A. J. Redd, D. E. Shumaker, R. Sydora, and J. Weiland. Comparisons and physics basis of tokamak transport models and turbulence simulations. *Physics of Plasmas*, 7(3):969–983, 2000.

[7] L. Fahrmeir, C. Heumann, R. Künstler, I. Pigeot, and G. Tutz. *Statistik: Der Weg zur Datenanalyse*. Springer, 2016.

[8] I.-G. Farcaș, T. Görler, H.-J. Bungartz, F. Jenko, and T. Neckel. Sensitivity-driven adaptive sparse stochastic approximations in plasma microinstability analysis, 2018.

[9] T. Görler, A. White, D. Told, F. Jenko, C. Holland, and T. Rhodes. A flux-matched gyrokinetic analysis of DIII-D L-mode turbulence. *Physics of Plasmas*, 21(12):122307, 2014.

[10] M. Griebel and J. Oettershagen. On tensor product approximation of analytic functions. *Journal of Approximation Theory*, 207:348 – 379, 2016.

[11] T. Görler, X. Lapillonne, S. Brunner, J. Chowdhury, T. Dannert, F. Jenko, B. F. McMillan, F. Merz, D. Told, and L. Villard. Nonlocal effects in gyrokinetic turbulence simulations using GENE. *Journal of Physics: Conference Series*, 260:012011, 2010.

[12] T. Görler, X. Lapillonne, S. Brunner, T. Dannert, F. Jenko, F. Merz, and D. Told. The global version of the gyrokinetic turbulence code gene. *Journal of Computational Physics*, 230(18):7053 – 7071, 2011.

[13] J. M. Hammersley and D. C. Handscomb. *Monte Carlo Methods*. Springer Science & Business Media, 1964.

[14] Iter Organization. Iter - the way to new energy, 2019.

[15] L. Jofre, G. Geraci, H. Fairbanks, A. Doostan, and G. Iaccarino. Exploiting multi-fidelity strategies to quantify uncertainty in irradiated particle-laden turbulent flow. 2018.

[16] J. A. Krommes. The gyrokinetic description of microturbulence in magnetized plasmas. *Annual Review of Fluid Mechanics*, 44(1):175–201, 2012.

[17] C. Lemieux. *Monte Carlo and Quasi-Monte Carlo Sampling*. Springer, New York, NY, USA, 2009.

[18] S. Li, H. Jiang, Z. Ren, and C. Xu. Optimal tracking for a divergent-type parabolic pde system in current profile control. *Abstract and Applied Analysis*, 2014:1–8, 2014.

[19] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[20] M. D. Morris, T. J. Mitchell, and D. Ylvisaker. Bayesian design and analysis of computer experiments: Use of derivatives in surface prediction. *Technometrics*, 35(3):243–255, 1993.

[21] B. L. Nelson. On control variate estimators. *Computers & Operations Research*, 14(3):219 – 225, 1987.

[22] L. W.-T. Ng and M. Eldred. Multifidelity uncertainty quantification using non-intrusive polynomial chaos and stochastic collocation. In *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference 20th AIAA/ASME/AHS Adaptive Structures Conference 14th AIAA*, page 1852, 2012.

[23] L. W. T. Ng and K. E. Willcox. Multifidelity approaches for optimization under uncertainty. *International Journal for Numerical Methods in Engineering*, 100(10):746–772, 2014.

[24] R. Pasupathy, B. Schmeiser, M. R. Taaffe, and J. Wang. Control-variate estimation using estimated control means. *Iie Transactions*, 44:381–385, 2012.

[25] B. Peherstorfer, K. Willcox, and M. Gunzburger. Optimal model management for multifidelity Monte Carlo estimation. *SIAM Journal on Scientific Computing*, 38(5):A3163–A3194, 2016.

[26] B. Peherstorfer, K. Willcox, and M. Gunzburger. Survey of multifidelity methods in uncertainty propagation, inference, and optimization. *SIAM Review*, 60(3):550–591, 2018.

[27] D. M. Pflüger. *Spatially Adaptive Sparse Grids for High-Dimensional Problems*. PhD thesis, Technische Universität München, 2010.

[28] M. J. Püschel. *Electromagnetic effects in gyrokinetic simulations of plasma turbulence*. PhD thesis, Westfälische Wilhelms-Universität Münster, 2009.

[29] E. Qian, B. Peherstorfer, D. O'Malley, V. Vesselinov, and K. Willcox. Multifidelity Monte Carlo estimation of variance and sensitivity indices. *SIAM/ASA Journal on Uncertainty Quantification*, 6(2):683–706, 2018.

[30] F. Riva, L. Milanese, and P. Ricci. Uncertainty propagation by using spectral methods: A practical application to a two-dimensional turbulence fluid model. *Physics of Plasmas*, 24(10):102302, 2017.

[31] J. L. Rodgers and W. A. Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42(1):59–66, 1988.

[32] R. C. Smith. *Uncertainty Quantification: Theory, Implementation, and Applications*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2013.

[33] A. Speight. A multilevel approach to control variates. *Journal of Computational Finance*, 12, 2009.

[34] A. M. Stuart. Uncertainty quantification in bayesian inversion. *ICM2014. Invited Lecture*, 1279, 2014.

[35] T. Sullivan. *Introduction to Uncertainty Quantification*, volume 63. Springer International Publishing, 2015.

[36] S. Surjanovic and D. Bingham. Borehole function, 2017.

[37] N. Thomopoulos. *Essentials of Monte Carlo simulation: Statistical methods for building simulation models*. Springer, 2013.

[38] D. Told, F. Jenko, T. Görler, F. J. Casson, and E. Fable. Characterizing turbulent transport in ASDEX Upgrade L-mode plasmas via nonlinear gyrokinetic simulations. *Physics of Plasmas*, 20(12):122312, 2013.

[39] P. Vaezi and C. Holland. An improved approach to uncertainty quantification for plasma turbulence validation studies. *Fusion Science and Technology*, 74(1-2):77–88, 2018.

[40] P. Vaezi, C. Holland, S. C. Thakur, and G. R. Tynan. Validation study of a drift-wave turbulence model for CSDX linear plasma device. *Physics of Plasmas*, 24(9):092310, 2017.

[41] K. Wiesemann. A short introduction to plasma physics, 2014.

[42] S. Xiong, P. Z. G. Qian, and C. F. J. Wu. Sequential design and analysis of high-accuracy and low-accuracy computer codes. *Technometrics*, 55(1):37–46, 2013.