High-Dimensional Uncertainty Quantification of Fluid-Structure Interaction

Ionuț-Gabriel Farcaș
Computational Science and Engineering
(Int. Master’s Program)
Technische Universität München

Master’s Thesis

High-Dimensional Uncertainty Quantification of Fluid-Structure Interaction

Author: Ionuț-Gabriel Farcaș
1st examiner: Prof. Dr. Hans-Joachim Bungartz
2nd examiner: Prof. Dr. Miriam Mehl
Assistant advisor: Dipl.-Math. Benjamin Uekermann
Thesis handed in on: October 19, 2015
Acknowledgments

First of all, I would like to thank my supervisor, Benjamin Uekermann, who not only suggested the topic of this project, but also offered me constant guidance and mentorship throughout its development. Moreover, I would like to thank Prof. Hans-Joachim Bungartz for allowing me to carry out this work at the chair of Scientific Computing.

Many thanks to Fabian Franzelin from Universität Stuttgart, whose insights and feedback played a major role in tackling some of the challenges of this work. Moreover, I would also like to express my appreciation to Alfredo Parra Hinojosa and Friedrich Menhorn for their valuable suggestions.

Above all, I would like to thank my family and my girlfriend Lavinia, for their constant support, encouragement, and warm thoughts. Without them, I would not be where I am today.

Last but not least, I would like to sincerely thank Deutscher Akademischer Austauschdienst (DAAD) for offering me the financial support necessary to study in Germany and to carry out my Master’s studies at Technische Universität München.
Abstract

This work combines an example of a multi-physics application, namely fluid-structure interaction, with uncertainty quantification. This is motivated by the goal of obtaining more realistic simulations via a comprehensive understanding of the impact of input uncertainties in the outcomes. Furthermore, performing multi-physics simulations is a must, as most of the real-world problems are complex, comprising components with individual physical descriptions. Due to the challenges posed by both domains, their combination leads to a multi-challenge, which implies that it is imperative to find suitable methods to deal with multi-physics, multi-dimensionality, and multi-core problems.

In the current work the multi-challenge is tackled by employing high performance computing and sophisticated mathematics. The simulations are performed on multi-core architectures using two levels of parallelism, one at the level of an individual run and the other at the level of the entire set of runs. The high dimensional problems are approached via the pseudo-spectral approach in combination with sparse grids interpolation. This methodology is non-intrusive and once the simulation results are available, they can be reused. The post-processing step comprises statistics estimation, probability density function estimation, and variance based sensitivity analysis. The testing is performed using two simulation scenarios. For both scenarios, uncertainty is modelled in five input parameters: density and dynamic viscosity of the fluid, density of the structure, Young’s modulus, and Poisson’s ratio.

The first scenario is a two dimensional wall-mounted flap and it is tested for stochastic dimensionality one, two, and five. The pseudo-spectral approach is validated using the Monte Carlo algorithm for dimensionality one and five. The post-processing step of the first two settings shows that the uncertainty in two out of the five input parameters is the most significant for the resulting uncertainty. Moreover, the analysis of the five dimensional case reveals that the same two parameters together with their interaction are the most significant for the outcomes, implying that the five dimensional problem could be reduced to a two dimensional one. The second scenario represents an established fluid-structure interaction benchmark, for which the five dimensional stochastic setting is simulated. Employing a sparse grid of level 3 for the interior domain and one of level 0 for the boundary, a total of 103 points is used. All simulations were performed on the MAC Cluster from the Munich Center of Advanced Computing in a time span of eight days, employing between 10 and 18 nodes. The total compute hours were $24h \times 103$, with 16 cores/simulation. The post-processing reveals that uncertainty in all five input parameters has a significant impact in the movement of the structure. Moreover, from the sensitivity analysis, it results that uncertainty in all input parameters is relevant in the outcomes, meaning that in this scenario, the stochastic dimensionality cannot be reduced.
## Contents

Abstract i

1 Introduction 1  
1.1 Problem Statement ........................................... 2  
1.2 The Multi-Challenge ......................................... 3  
1.3 Related Work .................................................. 4  

2 Fluid-Structure Interaction 7  
2.1 Theoretical Background ....................................... 7  
2.2 Modelling ....................................................... 9  
2.3 General Description of the Multi-Physics Code ............ 11  
2.4 General Description of the Coupling Library ............ 13  

3 Forward Propagation of Uncertainty 15  
3.1 Theoretical Background ....................................... 15  
3.2 Monte Carlo Sampling ......................................... 17  
   3.2.1 Algorithm .................................................. 17  
   3.2.2 Convergence and Properties ............................... 17  
3.3 Spectral Methods for Uncertainty Assessment ............ 18  
   3.3.1 Approximations via Polynomial Expansions .......... 19  
   3.3.2 Pseudo-spectral Approach ................................. 21  
   3.3.3 Convergence and Properties .............................. 23  
3.4 Discrete Projections using Sparse grids ................. 24  
   3.4.1 Interpolation on Sparse Grids ......................... 24  
   3.4.2 Pseudo-spectral Approach using Sparse Grids ....... 28  
   3.4.3 Sparse Grids Toolbox SG++ ............................ 31  
3.5 Probability Density Function Estimation ................... 31  
3.6 Variance Based Sensitivity Analysis ....................... 33  

4 Implementation Details 37  
4.1 General Description ........................................... 37  
4.2 Pre-processing ................................................ 41  
4.3 Simulation ..................................................... 44  
4.4 Post-processing ............................................... 44  

5 Test Scenarios and Results 49  
5.1 Vertical Flap .................................................. 49  
   5.1.1 Deterministic Simulation ................................. 50  
   5.1.2 One Dimensional UQ Simulation ....................... 51
## Contents

5.1.3 Two Dimensional UQ Simulations .......................... 59
5.1.4 Five Dimensional UQ Simulation ........................ 66
5.2 FSI Benchmark Scenario ....................................... 70
  5.2.1 Deterministic Simulations ............................... 72
  5.2.2 Five Dimensional UQ Simulation ......................... 72

6 Conclusions ................................................. 77

Bibliography .................................................. 79
1 Introduction

Due to the recent advancements in computational science, the underlying numerical errors in simulations increasingly reach acceptable accuracy levels. This is also the case for multi-physics simulations. Nowadays, uncertainties in modelling multi-scale phenomena, such as fluid-structure, fluid-thermal, or fluid-acoustic interactions have a larger effect on the accuracy of computational simulations than discretization errors ([55]). It is therefore important to systematically quantify and assess the impact of the uncertainty before any simulations are carried out.

The investigation of uncertainty within a system is carried out via uncertainty quantification (UQ). UQ is one of the newest scientific domains and stems from applied mathematics, evolving at a very high pace due to its extensive applicability in both research and engineering. UQ has received an increased amount of attention over the past years, especially in the context of complex systems. Even though many models have been successful in revealing quantitative connections between predictions and observations, their usage is constraint by the ability to assign accurate numerical values to the parameters of the underlying governing equation(s) ([59]). Within the current work, all uncertainty related problems are addressed in a forward approach, meaning that the uncertainty is modelled in the input physical parameters, being propagated through the respective system’s model, followed by its quantification at chosen output(s) of interest. This type of methodology is termed forward propagation of uncertainty.

Before going any further, it is important to point out that one needs to distinguish between stochastic models and UQ. Stochastic models are ordinary or partial differential equations in which one of the terms is a stochastic process, i.e. a random variable that evolves over time, typically white noise or Brownian motion. On the other hand, in UQ one deals with a deterministic model, in which coefficients or input parameters (physical parameters, initial or boundary conditions) are uncertain and typically modelled as random variables/fields. Some illustrative examples are the soil parameters and groundwater heights in geotechnical engineering, wind loads and snow loads in structural engineering, the amount of precipitation and evaporation in hydrology, etc.

The application field employed in this work is fluid-structure interaction (FSI). Broadly speaking, FSI problems address the interaction of one or more structures with an internal or surrounding fluid flow. They play a prominent role in many scientific fields, and pose many challenges due to their strong non-linearities and multidisciplinary nature, making FSI a complex domain by itself. Representative examples are blood flow simulations, flutter analysis of airplanes, wind turbines, etc.

The rest of this chapter is organized as follows: the next section is focused on the problem statement, followed by a description of the multi-challenge aspect of this work, together with the proposed solutions. Furthermore, Section 1.3 presents the related work. The chapter concludes with describing the structure of the rest of work.
1 Introduction

1.1 Problem Statement

This project was proposed at the chair of Scientific Computing, the Computer Science department of Technische Universität München, and its purpose is to investigate the impact of uncertainty in multi-physics applications. In other words, it aims to set up a proof of concept about UQ and one example of multi-physics application, i.e. FSI. The end goal is a comprehensive overview of UQ in FSI, which besides the uncertainty analysis, is aimed to comprise information such as which UQ methodology is most suitable for FSI - via testing different settings and versions of the algorithms, how the boundaries imposed by the complexity of applications can or cannot be overreached when coupling FSI with UQ, or how expensive a complete set-up is.

In the current work two FSI scenarios were employed, one termed vertical flap (c.f. Section 5.1) and the other representing an established FSI benchmark scenario (c.f. Section 5.2). The mathematical models for both scenarios depend on five physical parameters, which were considered to exhibit uncertainty either individually or in combination. In addition, all stochastic outcomes were compared to the deterministic counterpart. From the two scenarios, the vertical flap is less complex than the other and serves as a test application in which uncertainty was comprehensively investigated within one, two, and five dimensional scenarios. On the other hand, the FSI benchmark was analysed by running only one deterministic and one stochastic simulation. Besides its high complexity, another reason for running one UQ simulation is due to the fact that as it represents an FSI benchmark, it can be viewed as a surrogate for real-world FSI applications, where the test scenarios are usually run only for specific parts or parameters of the system, known beforehand to be susceptible of exhibiting uncertainty. The employed UQ methodology includes a classical Monte Carlo sampling technique, used as a validation tool, and one methodology based on spectral expansions and the pseudo-spectral approach, which due to its superiority in small to moderate dimensionality, represents the core method of this work. Both methods were implemented for one and several dimensions, having the built-in capability to treat two types of random inputs, i.e. normally and uniformly distributed. Their description and properties are presented in Chapter 3.

For the first scenario, the effects of uncertainty were analysed at a (fixed) point of the elastic structure and two points from the fluid domain. Moreover, for dimensionality one and two, uncertainty can be analysed in the entire FSI domain. For the second scenario, the investigated output was the tip of the structure. The post-processing step for both scenarios comprised a broad spectrum of tools geared towards a comprehensive understanding of the effects of uncertainty. The aim for the vertical flap was to analyse all possible one-dimensional UQ scenarios from a statistical and probabilistic point of view, followed by two dimensional combinations, chosen based on the results from the univariate case, and the five dimensional UQ scenario, comprising uncertainty in all input parameters. On the other hand, for the benchmark scenario, only the five dimensional UQ setting was tested. Besides the statistical evaluation of the output, a global sensitivity analysis was performed for all multi-dimensional cases, with a view to unravelling how the one-dimensional constituents and their combination affect the total resulted variance. In a practical, complex applications, this analysis can offer crucial insight, such as whether one or a combination of parameters from the entire set contributes dominantly to the overall resulted uncertainty. In this way, the total number required simulations can be substantially reduced.
1.2 The Multi-Challenge

The challenges tackled by the present work are grouped under the umbrella of the *multi-challenge*. In the underlying context, it comprises the *multi-physics* and *multi-domain* aspects of FSI, together with the *multi-dimensional* component of UQ, and the *multi-core* dimension used to address the entire problem in a high performance computing (HPC) setting. Conceptually, a multi-physical system consists of more than one component governed by its own principle(s) for evolution or equilibrium, typically expressed as conservation or constitutive laws ([32]). In the current work, the simulated FSI scenarios represent multi-physical systems which comprise the following constitutive components: the fluid, the structure, and their interaction. The multi-domain component is a property of the FSI simulations too, and it is due to the employment of different domains for modelling the fluids and solids. Traditionally, the fluid modelling is realized in an *Eulerian* coordinate system, whereas solid mechanics is usually modelled within a *Lagrangian* environment ([3]). In order to realize their coupling, a separate physical domain was introduced, via employing the *Arbitrary Lagrangian-Eulerian* approach. Finally, the multi-dimensional component is reflected in the UQ environment. As the goal is to investigate uncertainty in FSI, simulations where more than one parameter is uncertain appear naturally, hence resulting in multi-dimensional UQ simulations. Except for the one-dimensional UQ scenarios, the multi-physics, multi-domain and multi-dimensional challenges are inherent in every simulation. In order to address them, HPC, i.e. *multi-core architectures* were employed for performing the simulations. The multi-core aspect constitutes a challenge as well, due to the intrinsic complexity of the FSI solvers. As a single FSI simulation is expensive and an UQ simulation comprises many simulations, the challenge consisted of finding both a suitable way to use the available resources for running all simulations and a suitable number of simulations such that the difference between the needed resources and available resources is as small as possible. Furthermore, to address the challenges from a mathematical point of view, the employed UQ methodology should exhibit *fast convergence* and *non-intrusiveness*, i.e. the algorithms should require a small number of simulations in order to converge, while the codes should be reusable and easy to adapt when the dimension changes. This motivates the employment of *sophisticated mathematics*, facilitating a comprehensive uncertainty analysis at a reasonable computational expense. For an overview of multi-physics simulations, the reader is referred to [32].

In order to perform HPC-suited runs of the FSI solvers, state of the art software packages were employed. The used multi-physics code is called *Alya*, which was designed to run efficiently on massively parallel machines. For the coupling of the multi-physics solver, the employed library was *preCICE*, which addresses the coupling issue from a *partitioned* point of view. More details with regard to these codes are outlined in Chapter 2.

For increased UQ dimensionality, the *curse of dimensionality* occurs. To address it, *sparse grids* were employed in the following way: the multi-dimensional solution was interpolated on a sparse grid and afterwards, the result was combined with one dimensional quadrature to efficiently perform multi-dimensional numerical integration. The software providing the sparse grids capabilities was the SG++ library. The description of sparse grids and SG++ is provided in Section 3.4.
1 Introduction

1.3 Related Work

The idea of coupling UQ and FSI is contemporary with the establishment of UQ as a de facto branch of computational science, together with the recent improvements in computing, which lead to smaller and smaller simulation errors, requiring more attention for understanding the impacts of uncertainty. That is why in the past decade, much research involving UQ coupled with FSI was made.

The authors in [24] focused on the FSI and UQ combination from an engineering perspective, by simulating uncertainties in the aerodynamic, structural, material, and thermal variables of a combustor liner. The FSI part consisted of the coupling between a computational fluid code, modelled via the three-dimensional, unsteady, compressible Navier-Stokes equations, coupled with a $k - \omega$ SST turbulence model, and a finite element structural analysis code. In the UQ environment, two methods were employed. For computationally in-expensive runs, they used the Monte Carlo method, whereas for the compute-intensive simulations, Monte Carlo was again employed, but in a different setting, consisting of a second-order response surface model together with a Box-Behnken design experiment. The quantities of interest were the stresses and their variations evaluated at the critical points of the liner. In post-processing, the cumulative distribution functions and sensitivity factors were outlined, which revealed that the inlet and exit temperatures have great influence on the hoop stress. Moreover, it resulted that for prescribed values of inlet and exit temperatures, the Reynolds number, coefficient of thermal expansion, gas emissivity and absorptivity, and the thermal conductivity of the material have similar impact on the hoop stresses.

In [58], the authors treated uncertainties in a single-degree of freedom stall flutter model, with torsional oscillations. The underlying motivation stemmed for the fact that upon analysing the deterministic response, it was revealed that the problem was sensitive to variations in structural natural frequency and structural non-linearity. Moreover, according to the authors, this paper was the first that attempted to treat the stall flutter problem in an uncertain framework. The employed UQ methodology included Monte Carlo sampling, which served as a comparison tool with the results obtained after employing arbitrary polynomial chaos methodology. By the latter, the authors refer to a generalized polynomial chaos methodology, based on Gram-Schmidt orthogonalization process. The simulations revealed that the respective system is sensitive to the considered input uncertainties. However, for limit cycle oscillations, the accuracy of the polynomial chaos approach decreased with time. Similar work, involving Gram-Schmidt based arbitrary polynomial chaos was realized in [55]. The problem of long-term periodic limit cycle oscillations (LCO) was treated in [57], where the improvement came by employing a methodology called probabilistic collocations for LCO (PCLCO). In this method, probabilistic collocations are applied to a time-independent parametrization of the deterministic realizations of the periodic response for different parameter values. LCO are parametrized by the frequency, relative phase, amplitude, a reference value, and the normalized period. However, even though the method proved successful when applied to a two-degree-of-freedom airfoil flutter model, it was employed only to problems with a single random variable and one main frequency. The same authors treated the problem of UQ applied to a linear piston problem ([37]), by employing a two-step approach with chaos collocations. It comprised a sensitivity analysis step, which was applied to efficiently narrow down the set of multiple
uncertain parameters to one element that had the largest influence. Thereafter, chaos collocation was employed to obtain the stochastic output of the problem. Moreover, the results obtained via collocations were compared to the ones realized via Monte Carlo simulations, with a focus on comparing their efficiency. In the end, a speed-up of a factor of 100 was obtained, compared to a fully UQ problem.

In [56], the authors focused on UQ in FSI unsteady oscillatory problems, using a simplex elements stochastic collocation approach, based on Newton-Cotes quadrature in simplex elements. To enable efficient time-dependent UQ simulations, they employed two methodologies. The first was represented by a parametrization in terms of the time-independent functionals, similar to [57], but followed by a second UQ formulation, aimed at achieving a constant accuracy in time with a constant number of samples, using a constant phase for the oscillatory samples. Therefore, the resulting formulation was not subject to a parametrization error and it can resolve time-dependent functionals that occur in transient behaviour.

In a different approach, the authors in [29], treated the problem of UQ in FSI using an adaptive sparse grid collocation method. The UQ methodology was applied to a prototype problem, i.e. a single oscillator exited by means of an interaction stochastic force corresponding to the Morison formula. In this test scenario, the collocations method, based on Lagrange interpolation polynomials, approximated the solution in the stochastic space, even in the presence of non-linearities. As comparison tools, Monte Carlo method and non-adaptive collocation method were employed.

In [14], the authors addressed the problem of uncertainty propagation in a multi-physics system with multiple component models coupled together through network coupling interfaces. The approach towards uncertainty was based on exploring the structure of the network coupled multi-physics system to efficiently construct a polynomial surrogate of the output, rather than collecting all uncertainties into a single set and treat the system as a black box. In this way, the solution was treated as a composite function, using a non-linear elimination strategy, where the model outputs were functions of the coupling terms, which, again, were functions of the uncertain parameters. The employed test scenario was an idealized model of a nuclear reactor. The non-linear elimination led to a smaller system, where a stochastic Galerkin procedure together with a Newton iteration were applied in order to compute the coefficients of a surrogate response surface that approximates the coupling terms as a polynomial of the input parameters. However, the approach towards creating the quadrature grid was based on a tensor product rule, hence, being appropriate only when the number of input parameters of the full system is small.

The rest of the work is organized in four major parts. The first one, comprising the next two chapters, consists of an introduction to the theoretical aspects of FSI and UQ, as well as an overview to the employed simulation tools and algorithms. The first chapter aims to overview the relevant theory for fluids and solids modelling, hence, providing the reader with a general introduction to FSI. Besides the mathematical modelling, the multi-physics and multi-domain aspects of FSI are emphasized, as well as the employed simulation software, i.e. the multi-physics code Alya and the coupling library preCICE. Afterwards, the UQ part of the current work is described, in Chapter 3. This chapter provides a description of the employed algorithms, i.e. Monte Carlo sampling and spectral polynomial expansions with the pseudo-spectral approach, with a focus on the latter, which is
the core method of this work. Moreover, as multi-dimensional UQ simulations suffer from the *curse of dimensionality*, the employed solution is presented in this chapter too, which consists of interpolation on sparse grids. The sparse grids functionality is provided by the library SG++. Even more, aspects related to the UQ post-processing step are described, such as the probability density function estimation and sensitivity analysis. Both aim to aid the understanding of the resulting uncertainty by giving a probabilistic description of the output, and describing how the total resulted uncertainty can be apportioned to the uncertainty in its inputs, respectively.

The second part describes the UQ implementation, where the reader is provided with an overview of the implemented code. The overview emphasizes the three major constitutive blocks: pre-processing, simulation, and post-processing, together with how it interacts with the application code.

After the reader has gained an understanding of the theory, employed software and implementation, the following part describes the obtained results, after applying the UQ code in the two simulation scenarios. Finally, the last chapter focuses on summary and conclusions.
2 Fluid-Structure Interaction

This chapter provides an overview of the FSI segment of this work. The first two sections describe the theoretical (Section 2.1) and modelling (Section 2.2) aspects of the field, focusing on its multi-physics facet, mathematical description of fluids and structures, as well as the approaches used for solving the resulting coupled problem. The last two sections describe the employed codes, i.e. the multi-physics package Alya and the coupling library preCICE, by providing their general overview, as well as some important features, relevant for the present work. Before going any further, it is important to remark that within this project, by FSI it is meant the interaction of elastic bodies with a laminar, isothermal, incompressible, Newtonian fluid flow. The reader is assumed to have basic knowledge of fluid and solid mechanics in one and multiple dimensions. As a comprehensive description of FSI is beyond the scope of the current work, the following overview focuses on summarizing the relevant theory, with literature references being listed for further reading.

2.1 Theoretical Background

Traditionally, fluid flows on one hand and elasticity together with solid mechanics on the other hand, are two separate scientific branches. Their coupling has been integrated in the framework of FSI, which besides these two components, it includes their interaction as well. Due to the tremendous advancements in computing from recent years, FSI has become a well established branch of computational science. Although simulations have been done since the first programmable computers appeared, complex problems were almost impossible to simulate, due to the high demand of computational resources. For a detailed overview of computational FSI, the reader is referred to [6] and [12].

Within the current work, the modelling of fluids and structures was done at a macroscopic, continuous level, in two spatial dimensions. In order to derive a mathematical model for an FSI problem, the descriptions of the fluid, structure, and their interaction must be undertaken. As they represent three different problems, the first task is to outline the corresponding coordinate systems. In solid mechanics, two different coordinate systems are usually used, namely the Lagrangian and the Eulerian coordinate systems. Briefly, in Lagrangian coordinates, a specific material point and its properties are observed over time, whereas in the Eulerian perspective, the point is fixed and its properties are observed as the time evolves. Their comparison is depicted in Figure 2.1. In order to achieve their coupling, the arbitrary Lagrangian-Eulerian (ALE) point of view was introduced ([16]). In the following, the employed mathematical terminology is presented.

Let $\Omega$ denote the reference, undeformed, or material configuration, with $\hat{\omega} \in \hat{\Omega}$ denoting a generic material point, whereas $\Omega(t)$ symbolizes the current, deformed, or spatial configuration, with $\omega \in \Omega(t)$ being a generic spatial point. Moreover, $\Omega_{t=0}$ is the so called initial configuration. In addition, $f(t, \omega)$ and $\hat{f}(t, \hat{\omega})$ denote a property in the material and spatial
domain, respectively. It is important to remark that the modelling is carried out from the perspective of an observer situated in the spatial domain. Often, $\hat{\Omega} = \Omega_{t=0}$. Moreover, the reference configuration $\hat{\Omega} \in \mathbb{R}^d$ is assumed to be open and connected ([3]).

As the Lagrangian domain is fixed, the correspondence between a material point in the reference and current configuration is realized via a mapping

$$x : \hat{\Omega} \times \mathbb{R} \rightarrow \Omega(t) \times \mathbb{R}, x(\hat{\omega}, t) = (\omega, t)$$

(2.1)

Note that the time variable is the same in both coordinate systems, only the spatial variables being changed. This mapping is required to be continuously differentiable, invertible, and orientation-preserving, meaning that its Jacobian determinant, denoted by $J_x$, must be strictly positive, i.e. $J_x > 0$ ([3]). The last requirement implies that a material point can be subjected only to small deformations. That is why the Lagrangian perspective is typically adopted in structural mechanics. Due to the fixed reference frame, its advantages include the implicit treatment of moving boundaries and moreover it enables the easy association of a property history to each material point. An important relation used for modelling in a Lagrangian framework is termed total or material derivative and reads as follows, assuming that the underlying property is scalar:

$$\frac{d\hat{f}}{dt} = \partial \hat{f} \partial t$$

(2.2)

In words, it states that the time evolution of a Lagrangian property $\hat{f}$ is equal to the local time change of that property ([3]). It holds analogously for vectorial case.

On the other hand, in Eulerian coordinates the frame of reference is identical to the spatial domain. This implies that no mapping is needed to characterize the current position of a material point. However, in order to relate a material property $f$ with a property from the current configuration $\omega$ at the same spatial position $\omega$, the mapping $x(\hat{\omega}, t)$ is used as:

$$\hat{f}(\hat{\omega}, t) = f(x(\hat{\omega}, t)) = f(\omega, t) \circ x(\hat{\omega}, t).$$

(2.3)

Assuming for simplicity that the properties are scalars as in the Lagrangian perspective case, their change over time results in the material derivative in the Eulerian framework, which reads as:

$$\frac{df}{dt} = \frac{\partial f}{\partial t} + (v \cdot \Delta) f.$$ 

(2.4)

In words, it means that the total change of a property is yielded by its time derivative, plus the convective transport of neighbouring material with material velocity $v$. The convective term appears due to the change of the property in space in Eulerian coordinates.
2.2 Modelling

These equations hold analogously for the vectorial case. The identity of spatial and reference domains implies that the Eulerian approach enables the treatment of arbitrary deformations. However, its main drawback is that it cannot describe the boundaries of the observed material. Because of this, it is usually employed in fluid mechanics.

In order to treat FSI as a whole, Lagrangian and Eulerian perspectives need to be used together. Being intrinsically different, one method to find a common denominator is the (ALE) approach. For a comprehensive description, the reader is referred to [16] and the references therein. Briefly, with ALE, a third frame of reference is considered, termed the referential domain, denoted by $\Omega_{ALE}$. Within this new domain, points of observation can move independently of the material, resulting in a mesh motion, additional to the material motion in the spatial domain. This leads to a convective velocity, denoted by $v_{ALE}$, which is the relative velocity between material and referential domain, as seen from the spatial coordinate system. One way to overcome the main limitation of the Eulerian approach is to employ the ALE approach instead. Hence, fluid mesh velocity is chosen such that at the interface of fluid and structure a Lagrangian behaviour is obtained, fitting the structural description. Even more, the distortions of the interface are balanced out by a suitable mesh smoothing procedure in the interior of the fluid domain ([16]). However, ALE is not suitable for arbitrary large deformations without re-meshing techniques ([41]). It is important to note that both Lagrangian and Eulerian approaches can be recovered from the ALE framework as special cases ([16]).

2.2 Modelling

In the context of fluid and solid mechanics, the underlying continuous mathematical models are derived from balance principles and conservation equations, meaning that the conservation of mass, momentum, angular momentum, and energy yield the continuous models. The conservation laws result from applying the Reynolds transport theorem in the underlying coordinate system, which states what the time derivative of an integral over a time-dependent volume is. In what follows, the mathematical framework for both fluid and solid mechanics is outlined without proofs (for more details, see [5, 8, 18]).

For fluid mechanics, the underlying conservation laws are obtained using the Reynolds transport theorem in Eulerian coordinates. As stated in the beginning of this chapter, in the current work fluids were considered to be isothermal and incompressible, hence the conservation laws are derived only for mass and momentum. The Reynolds theorem states that:

$$\frac{d}{dt} \int_{V(t)} f \, dV = \int_{V_c=V(t)} \left( \frac{\partial f}{\partial t} + (v \cdot \Delta)f + f(v \cdot \Delta) \right) dV, \quad (2.5)$$

where $V(t)$ is a time-dependent material volume, and $V_c$ is a fixed control volume with fixed control surface $S_c$. By applying this theorem for $f$=mass, and $f$=momentum the isothermal, incompressible Navier-Stokes equations are yielded, which in differential form read as:

$$\nabla \cdot v = 0 \quad (2.6)$$

and

$$\rho \frac{\partial v}{\partial t} + \rho (v \cdot \nabla)v - \nabla \cdot \sigma = \rho f, \quad (2.7)$$
where \( v \) denotes the velocity, \( \rho \) the density, \( \sigma \) is the Cauchy stress tensor, and \( f \) is the distributed volume force (e.g. gravity) ([5]). The first equation is known as the continuity equation and describes a divergence-free flow, while the second equation is called the momentum equation. These equations alone are not sufficient for having a unique solution to the flow problem. The set of equations describing fluids is closed when defining the following three important aspects: the constitutive law for the Cauchy stress tensor, initial conditions, and boundary conditions. For Newtonian fluids, i.e fluids characterized by a linear stress-strain relationship with a vanishing strain for zero stress, such as water or air, the usually employed constitutive law is the viscous flow with inner friction that reads as:

\[
\sigma = -\rho I + \mu (\nabla v + \nabla v^T),
\]

(2.8)

where \( \mu \) is the dynamic viscosity ([5, 18]). As initial conditions, the prescribed quantity is the flow velocity \( v \). The boundary conditions are typically a combination of Dirichlet - for the kinematic variables, such as the velocity, and Neumann - for dynamic variables, such as the pressure, boundary conditions. Finally, a key quantity in fluid flows simulations is the Reynolds number, which is a dimensionless quantity defined as \( Re = \frac{lv}{\nu} = \frac{dv}{\nu} \), where \( l \) and \( v \) are the characteristic length and velocity, respectively, and \( \nu = \frac{\mu}{\rho} \) is the kinematic viscosity.

In the realm of solid mechanics, the derivations are similar to the ones for fluids, with the difference of deriving the conservation laws in Lagrangian coordinates. This implies that there are no convective terms. Hence, the momentum equation reads as:

\[
\rho \frac{\partial^2 u}{\partial t^2} = \nabla \cdot S + \rho f,
\]

(2.9)

where \( u \) denotes the displacement of the material, \( \rho \) the density of the material, \( S \) is the 2nd Piola-Kirchhoff stress tensor, and \( f \) denotes the body forces, such as gravity ([8]). In words, this equilibrium relation states that there must be an equilibrium of forces within the volume of a structure. The \( S \) stress tensor models the surface forces and it is assumed that the used material obeys a linear strain-stress relationship ([8]). The typically employed linearised constitutive relation expressed in tensor notation, reads as:

\[
S_{ij} = \lambda E_{kk}\delta_{ij} + 2\mu E_{ij},
\]

(2.10)

where \( \lambda \) and \( \mu \) are the Lamé constants and \( E \) is the strain tensor. Relation 2.10 is called the constitutive relations for a Saint-Venant-Kirchhoff material ([8]). The Lamé constants and material parameters, such as Young’s modulus \( E \) and Poisson’s ratio \( \nu \), are related as:

\[
E = \frac{\mu(3\lambda + 2\mu)}{\lambda + \mu}, \quad \nu = \frac{\lambda}{2(\lambda + \mu)}.
\]

(2.11)

The initial and boundary conditions are set in a similar manner to the case of the fluid modelling. The initial conditions are prescribed for the displacement and velocity, while the boundary conditions are prescribed for the displacement - of Dirichlet type, and for surface forces - of Neumann type.

After having the mathematical models for fluids and structure, the next step is their coupling. Denoting by \( \Omega_f \) the fluid domain, with boundary \( \Gamma_f = \partial\Omega_f \), and by \( \Omega_s \) the solid counterpart, with boundary \( \Gamma_s = \partial\Omega_s \), their coupling occurs at the common interface,
denoted by \( \Gamma_{fs} = \Gamma_f \cap \Gamma_s \), sometimes termed the wet surface. At the wet surface, kinematic and dynamic boundary conditions are enforced. The kinematic conditions impose the velocities to be equal at \( \Gamma_{fs} \), whereas the dynamic conditions enforce a balance of forces at the interface, expressed by the corresponding stress tensors ([3, 12]). Mathematically, these conditions read as:

\[
v_f = v_s \\
\sigma_f n = \sigma_s n,
\]

(2.12)

where \( n \) is a unit normal vector to the interface \( \Gamma_{fs} \).

2.3 General Description of the Multi-Physics Code

The multi-physics code employed throughout the current work is called Alya. Alya System - Large Scale Computational Mechanics\(^1\) is a Fortran-based simulation code developed since 2004 at Barcelona Supercomputing Center, in Spain, designed for multi-physics problems. Its capabilities comprise incompressible and compressible flows, non-linear solid mechanics, N-body collisions, etc. Moreover, its design includes two main features, which are strongly interconnected: first, it was specially designed for running with high efficiency on modern multi-core machines and second, it is capable of solving different physics, each with different characteristics. It is one code, designed directly in parallel, having the capability to be run under both Linux/Unix and Microsoft Windows operating systems. Furthermore, its constituents can be compiled individually.

Alya’s main characteristics include modular architecture, finite element (FEM) space discretization (based on non-structured meshes, read from text files), the capability of solving problems with mixed meshes, and internal, external, and hybrid parallelization. The constitutive Alya’s components are termed modules, services, and a kernel, where the modules were designed to solve one individual problem, services play the role of a toolbox that aids the modules and the kernel, and finally, the kernel is the core of Alya\(^2\). Furthermore, external parallelization is done via MPI\(^3\) - for communication, and METIS\(^4\) - for problem subdivision. Internally, Alya uses OpenMP\(^5\) directives to distribute the code loops amongst threads. Finally, the hybrid version uses both strategies at the same time, aimed especially for running on multi-core machines. While the space discretization is realized via FEM, for the time discretization, Alya employs finite differences.

In the current work, the employed modules are the one for incompressible flow, called nastin, and the one for non-linear solid mechanics, i.e. solidz. For tackling the multi-domain challenge, Alya employs the ALE approach. Moreover, among the available services, the one used in this work is called parall, being used for parallel simulations.

Alya’s set-up is done via several configuration files. To put this in perspective, consider that a simulation named sim is to be performed. In pre-processing, a series of files neces-

---

\(^1\)see https://www.bsc.es/computer-applications/alya-system for more details

\(^2\)see https://www.bsc.es/computer-applications/alya-system/alya-features for more details

\(^3\)see http://www.mpi-forum.org/ for more details

\(^4\)see http://glaros.dtc.umn.edu/gkhome/views/metis for more details

\(^5\)see http://openmp.org/wp/ for more details
sary for setting up the constitutive parts of Alya is required. As an example, if the nastin and solidz modules are employed, the configuration is realized as follows:

- **The kernel** - three mandatory files
  - `sim.dat` - general data, such as employed modules and services
  - `sim.dom.dat` - domain input data, i.e. mesh description
  - `sim.ker.dat` - kermod input data - when coupling different physics, the shared variables among modules are specified here

- **Modules**
  - `sim.nsi.dat` - input data for nastin
  - `sim.sld.dat` - input data for solidz

- **Services**
  For e.g. parall, no extra file is required - the characteristic information is specified inside all .dat files.

The generic configuration files are depicted in Figure 2.2. Note that the `sim.nsi.dat` and `sim.sld.dat` represent the mean for realizing Alya’s coupling with the UQ code.

![Alya input data files diagram](source: http://bsccase02.bsc.es/alya/index.html)

The end of an \( n \) times-steps simulation performed by employing the nastin and solidz modules has an output consisting of information comprising fluid’s pressure on \( x \) axes,
velocity on x and y axis, and displacement and velocity of the mesh, caused by the ALE approach, and the displacement of the structure. For each quantity, $n$ binary files are output. For their post-processing, Alya offers an external tool called *alya2pos*, used to transform the binary files into *Ensight* format.

In Alya’s documentation it is stated that the solution procedure consists of five main steps: *initialization*, followed by *pre-processing*, *simulation*, and *post-processing*, ended with a *finalization* step. It should be noted that Alya’s functionality comprises the coupling of its modules as well. However, in this work this is realized by employing *preCICE*, as it is described in the following section. For an overview of Alya’s recent advancements, the reader is referred to [52].

### 2.4 General Description of the Coupling Library

The software package used in the current work for coupling the modules of the multi-physics package is called *preCICE*[^7], which stands for *preCICE Code Interaction Coupling Environment*. Initiated at Technische Universität München, the project is currently developed jointly at Technische Universität München and Universität Stuttgart. Furthermore, it is part of the *SPPEXA ExaFSA* project[^8]. As main reference, the reader is referred to [22].

*preCICE* is a C++ based library which offers *partitioned coupling capabilities* to a wide range of solver, focusing on FSI, and a *geometry interface* for Cartesian grid solvers. The coupling is realized via four main functionality groups: *equation coupling*, *data mapping*, *communication*, and *geometry interface* group. Moreover, it provides an application programming interface (API) for C/C++ and Fortran, which consists of high-level library-like methods. Following its successful integration in the application code, the communication between the components of the application is realized in a fully parallel point-to-point manner, based on either TCP/IP or MPI ports. The configuration is done at runtime, via an XML file, where information regarding the data, meshes and geometry of meshes, participants[^9], communication, and coupling algorithms is specified. Note that a functionality called *watch-point* has been implemented in *preCICE* as well, which records the displacements and forces for a point in the structure domain, whose coordinates were specified by the user in the configuration file (see [22] for more details).

*preCICE* couples established codes, which are treated as *black boxes*. It is assumed that the simulation code has the logical skeleton outlined in Algorithm 1 ([22]). Even more, it suffices that the solver provides access before and/or after the enumerated steps ([22]). As described in Section 2.3, Alya’s solution procedure obeys the above structure, hence enabling the usage of *preCICE*.

It follows that the nastin and solidz modules were coupled by employing the partitioned approach. Moreover, in the current work the communication is realized by employing TCP/IP ports, whereas the data-mapping is performed in a *projection-based manner*, using a nearest neighbour approach. Two types of nearest neighbour projection mappings were

[^6]: [see http://www-vis.lbl.gov/NERSC/Software/ensight/doc/OnlineHelp/UM-C11.pdf for more details]
[^7]: [see http://www.precice.org/ for more details]
[^8]: [see https://ipvs.informatik.uni-stuttgart.de/SGS/EXAFSA/index.php for more details]
[^9]: in the *preCICE* context, a participant is one solver from the underlying multi-physics code
Algorithm 1 Necessary structure of a simulation code

1: initialization
2: while not finished do
3:    pre-processing
4:    solution
5:    post-processing
6: end while
7: finalization

used: a conservative mapping, i.e. a mapping which ensures that the sum of the data values in each spatial direction is equal on both sides, was used from the fluid domain to the solid domain, while a consistent mapping strategy was employed in the opposite direction. For more details about preCICE’s data-mapping capabilities, see [22]. In Figure 2.3, a generic overview of preCICE is depicted. For the latest advancements of preCICE, the reader is referred to [7, 11, 35].
3 Forward Propagation of Uncertainty

This chapter provides a detailed description of the UQ part of the current work. Its focus is on forward propagation of uncertainty, i.e. problems where system output(s) were propagated from uncertain inputs. It begins with an overview of probabilistic and statistics theory in Section 2.1, continuing with the description of the first employed algorithm in Section 3.2. Section 3.3 details the second stochastic algorithm, based on spectral expansions and discrete projections. The following section overviews the approach employed for tackling the challenge posed by multi-dimensional UQ problems. Finally, the last two sections describe two techniques used in post-processing: probability density function estimation via kernel density estimation in Section 3.5 and global sensitivity analysis in Section 3.6.

3.1 Theoretical Background

In this section, the mathematical framework used in the context of forward propagation of uncertainty is presented. It is mainly based on definitions and notations found in [1, 30, 47, 59, 60, 61]. Before going any further, it is assumed that the reader has a basic understanding of probability and measure theory, being familiar with concepts such as probability space, σ algebra, probability measure, continuous probability density function, continuous random variables, and of statistics, such as expectation and variance of a continuous random variable. For an overview of these concepts, please refer to [31].

Let \((\Omega, \mathcal{F}, \mu)\) denote an abstract probability space, where the set \(\Omega\) represents the sample space, \(\mathcal{F}\) is a σ-algebra defined on \(\Omega\), and \(\mu\) is a probability measure. A continuous random variable is denoted by \(U\) and represents a real valued mapping of the form:

\[
U : (\Omega, \mathcal{F}, \mu) \to (\mathbb{R}, \mathcal{B}(\mathbb{R})),
\]

where \(\mathcal{B}(\mathbb{R})\) is the Borel σ-algebra on \(\mathbb{R}\). Given a random variable \(U\), its expectation and variance are defined as:

\[
\mathbb{E}[U] = \int_\Omega U(\omega)\mu(d\omega), \quad \text{Var}[U] = \mathbb{E}[U^2] - \mathbb{E}[U]^2.
\]

(3.2)

Let \(L^2(\Omega, \mathcal{F}, \mu)\) denote a Hilbert space of real-valued square integrable random variables on \(\Omega\). The corresponding inner product is defined as \((\cdot, \cdot) : L^2(\Omega) \times L^2(\Omega) \to \mathbb{R}\),

\[
(U, V) = \mathbb{E}[UV] = \int_\Omega U(\omega)V(\omega)\mu(d\omega), \quad U, V \in L^2(\Omega, \mathcal{F}, \mu),
\]

(3.3)

with the associated norm:

\[
\|U\|_{L^2(\Omega)} = (U, U)^{\frac{1}{2}}.
\]

(3.4)
In what follows, $\xi$ denotes a single random variable on $\Omega$, whereas $\xi = (\xi_1, \ldots, \xi_d)^T$ denotes a $d$-dimensional collection of random variables. The distribution function associated to $\xi$ is usually denoted by $F_\xi$ and is defined as:

$$F_\xi = \mu(\xi \leq x), \ x \in \mathbb{R}. \quad (3.5)$$

Moreover, a continuous random variable has an associated pdf defined as $\rho(x) = \int_{-\infty}^{x} F_\xi(dt)$.

For everything that follows, by random variable it is meant continuous random variable. Given that the majority of algorithms in UQ are defined in terms of standard random variables, the generalization to multiple dimensions is easily done. For convenience, in what follows, by random variable it is meant continuous random variable.

The support for a standard uniform random variable $U$ is $[-1, 1]$, with the pdf $\rho_u = \frac{1}{2}$, whereas a general uniform is supported by $[a, b]$, $a < b$, having the associated pdf $\rho_u = \frac{1}{b-a}$. For the latter, $a$ is usually referred to as the left parameter, while $b$ is as the right parameter. On the other hand, for both standard and general normal random variable, the support is $\mathbb{R}$. However, they are identified by two defining parameters: the mean, denoted by $\mu$, and the standard deviation, denoted by $\sigma$. A standard normal $U_n$ has zero mean and unit standard deviation, being denoted as $U_n \sim \mathcal{N}(0, 1)$, having the associated pdf $\rho_n = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$. On the other hand, a generalized normal random variable is defined as $U_n \sim \mathcal{N}(\mu, \sigma^2)$, with the corresponding pdf $\rho_n = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$. Under the i.i.d. assumption, the generalization to multiple dimensions is easily done. For convenience, in everything that follows, by random variable it is meant continuous random variable.

Given that the majority of algorithms in UQ are defined in terms of standard random variables, a connection with generalized random variables is required. For the generalized uniform, the connection is simply realized via a linear transformation:

$$T : [-1, 1] \to [a, b], T(x) = \frac{b-a}{2} x + \frac{b+a}{2}. \quad (3.8)$$

On the other hand, the connection to a standard and generalized random variable is realized via the normal linear transform theorem ([33]):

$$\mu + \sigma \mathcal{N}(0, 1) = \mathcal{N}(\mu, \sigma^2). \quad (3.9)$$
3.2 Monte Carlo Sampling

The Monte Carlo family of algorithms is widely used within numerical mathematics, statistics, and UQ. Besides the simplest version termed Monte Carlo Sampling (MCS), another UQ-relevant variant is Markov Chain Monte Carlo (MCMC). Historically, the idea of this algorithm appeared in the 1940s, having John von Neumann, Stanislas Ulam, and Nicolas Metropolis as its inventors ([40]). However, for reasons that are detailed later in the chapter, in the current work it serves only as a comparison tool, used for validating the results obtained with the other employed method. Next, the one and multiple dimensions versions of the algorithm are presented, continuing with a description of its properties. For a more detailed description of Monte Carlo algorithms in UQ, please refer to [47].

3.2.1 Algorithm

In the first step, $n$ random samples are generated based on a prescribed pdf, where $n$ is user defined. Afterwards, each generated sample is input in the numerical solver of the system and the resulting numerical equation is solved. Using UQ terminology, this translates as all realizations of the stochastic input are forwardly propagated through the system’s equation. Note that after the stochastic input is plugged into the model of the system, the problem becomes deterministic ([59]). At the end of each simulation, observables, which in UQ are termed quantity of interests (QoI), are retrieved. Finally, after all samples are propagated, the last step consists of post-processing. Typically, it comprises the statistical evaluation of the QoI and/or the estimation of their corresponding pdf. It is important to note that provided the i.i.d. nature of the input parameters, the extra effort needed to transform the univariate algorithm into a multi-dimensional version consists of only generating the additional random variables for all extra dimensions. The steps for the one dimensional case were summarized in Algorithm 2.

**Algorithm 2 Monte Carlo sampling**

<table>
<thead>
<tr>
<th>Require: $n$ value</th>
<th>user defined number representing the number of samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Generate $n$ random samples $\xi_i$, $i = 0 \ldots n - 1$</td>
<td>based on an a prescribed input pdf</td>
</tr>
<tr>
<td>2: for $i \leftarrow 0$ to $n - 1$ do</td>
<td></td>
</tr>
<tr>
<td>3: plug $\xi_i$ in and solve the underlying numerical equation</td>
<td></td>
</tr>
<tr>
<td>4: end for</td>
<td></td>
</tr>
<tr>
<td>5: Estimate the statistics for the chosen QoI, e.g.</td>
<td></td>
</tr>
<tr>
<td>6: expectation: $E[U(t, \xi)] = \int \rho(\omega) d\omega \approx 1/n \sum_{i=1}^{n} U^{(i)}(t, \xi^{(i)})$</td>
<td></td>
</tr>
<tr>
<td>7: variance: $Var[U(t, \xi)] = E[U(t, \xi) - E[U(t, \xi)]]^2 \approx 1/n \left( \sum_{i=1}^{n} \left( U^{(i)}(t, \xi^{(i)}) \right)^2 - \frac{1}{n} \sum_{i=1}^{n} U^{(i)}(t, \xi^{(i)})^2 \right)$</td>
<td></td>
</tr>
<tr>
<td>8: Estimate the pdf for the chosen QoI</td>
<td></td>
</tr>
</tbody>
</table>

3.2.2 Convergence and Properties

The main characteristics of MCS are simplicity to understand, use, and implement. Moreover, given the little effort needed to extend it from one to multiple dimensions, MCS is
also robust. The latter together with the fact that it treats the underlying numerical solver as a black box, make the algorithm non-intrusive, i.e. it relies on a deterministic version of the system’s numerical solver, not on a stochastic surrogate. Moreover, once an existing code is available, it can be easily reused ([47]). Furthermore, MCS is independent from the choice of input distribution and in addition, the statistical evaluation is generic, requiring only the sum and sum of squares of the corresponding QoI. The propagation of the stochastic variables results in independent calls of the system’s numerical solver, making the algorithm embarrassingly parallel, hence, well suited for HPC ([59]).

However, MCS works in a generic way, without exploring the frequently exhibited smooth dependence of the solution on the stochastic parameters ([59]). In addition, the algorithm relies on the law of large numbers ([47]). The last two properties imply that a large number of samples is required to obtain accurate results. The mean value typically converges as $O\left(\frac{1}{\sqrt{n}}\right)$ ([47]), where $n$ is the number of samples. If the employed numerical solver is computationally expensive, propagating a large number of samples is not realistic. Even worse, as the number of samples increases, so do the memory requirements, thus making the algorithm memory expensive as well. Throughout the years, efforts have been made to improve the convergence rate of MCS. Illustrative examples are the Latin hypercube sampling ([38]) and Quasi Monte Carlo (QMC) methods ([20]), which achieve a convergence rate of $O\left(\frac{1}{n}\right)$, for an input of size $n$.

Finally, the numerical error of MCS is presented. As it requires the numerical solver of the underlying system, MCS inherits its associated numerical error. Moreover, the sampling strategy introduces a numerical error as well. Denoting a generic numerical error with $\epsilon$ and assigning appropriate indices, it follows that the overall MCS error is:

$$\epsilon_{MCS} = \epsilon_N + \epsilon_S,$$

where $N$ denotes the numerical solver and $S$ sampling.

### 3.3 Spectral Methods for Uncertainty Assessment

A family of methods better suited to cope with complex UQ problems is termed spectral methods for uncertainty assessment, which includes: stochastic collocations (SC), stochastic Galerkin (SG), and the discrete projections method. The latter is employed in the current project and it constitutes the principal UQ methodology. The relevant references are [2, 19, 30, 34, 47, 53, 59, 60, 61, 62].

The key aspect of spectral expansions is that they exploit the frequently exhibited smooth dependence of the solution on the inputs, leading to an accurate approximation obtained at a low computational cost. The choice of the basis functions terms the expansions either polynomial chaos (PC) or generalized polynomial chaos (gPC) expansions. The former was coined in the seminal work of Norbert Wiener ([54]) from 1938, who employed a Hermite polynomial basis for constructing a physical theory of chaos ([47]). The method was brought back into the scientific and engineering communities’ attention by the work of Ghanem and Spanos in [23]. However, in the context of UQ, PC or gPC terminology is a misnomer, since the underlying systems are not typically chaotic. To the degree possible, in the current work the alternatively used term is spectral expansions. In the following subsection,
the theory behind the employed polynomial expansions is presented. Next, the *pseudo-spectral approach* used for calculating the expansion coefficients is described, followed by an overview of properties and convergence analysis, in Section 3.3.3.

### 3.3.1 Approximations via Polynomial Expansions

Consider an infinite sequence of random variables \( \{\xi_k(\omega)\}_{k=1}^{\infty} \) defined on the sample space \( \Omega \). Additionally, consider a sequence \( \{P_k\}_{k=1}^{\infty} \) of orthogonal polynomials, i.e.

\[
(P_i, P_j) = c \delta_{ij}, \tag{3.11}
\]

where \( c \) is a constant and \( \delta_{ij} \) denotes the Kronecker delta, defined as:

\[
\delta_{ij} = \begin{cases} 
1, & \text{if } i = j, \\
0, & \text{if } i \neq j.
\end{cases} \tag{3.12}
\]

Employing a special case of Cameron-Martin theorem (see [13] for more details), every second order or finite variance random variable admits an expansion of the form:

\[
U(\omega) = \sum_{k=0}^{\infty} c_k \Psi_k(\xi), \tag{3.13}
\]

where \( \Psi_k \) is an \( \infty \)-variate orthogonal polynomial ([47]). In practice, it makes sense to consider a finite sequence of random variables and a limited number of expansion terms. Hence, for a \( d \)-dimensional set of random variables \( \{\xi_k(\omega)\}_{k=1}^{d} \) and \( p \) interaction terms, Equation 3.13 becomes:

\[
U^K(\omega) = \sum_{k=0}^{K} c_k \Psi_k(\xi_1, \ldots, \xi_d), \tag{3.14}
\]

where \( K = \frac{(d+p)!}{d!p!} \). Using the equivalence between the representation of a random process \( U(t, \xi), t \in [0, T] \) in the abstract probability space \( (\Omega, \mathcal{F}, \mu) \) and the image space \( (\Gamma, B(\Gamma), F_\xi(dx)) \), the finite dimensional expansion 3.14 becomes:

\[
U^K(t, \xi) = \sum_{k=0}^{K} c_k(t) \Psi_k(\xi). \tag{3.15}
\]

Thus, the deterministic part \( c_k(t) \) is separated from the stochastic part represented by \( \Psi_k(\xi) \). It follows that in order to obtain the representation of a random process, one needs to specify appropriate basis functions \( \Psi_k \) and constraints used to determine the coefficients. In the current work the polynomials were considered to be globally defined, an alternative being piecewise defined polynomials ([47]). As an observation, note that if \( \Psi_i(\xi) = \delta(\xi - \xi_i) \), where \( \xi_i \) is an isolated random event and \( \delta \) is Dirac’s delta function, MCS results as a particular case of the above described methodology ([60]).

In the one-dimensional case, the employed polynomials are denoted by \( \phi_k \) and indexed so that \( \phi_0 = 1 \). They are orthogonal with respect to the density \( \rho_\xi \), i.e.:

\[
E[\phi_i(\xi)\phi_j(\xi)] = \delta_{ij} \gamma_i, \tag{3.16}
\]
where $\gamma_i$ is termed the *normalization factor*. This property together with Equation 3.15 provide an easy way to compute statistics such as mean and variance using only the expansion coefficients, as following ([47]):

\[
E[U^K(t, \xi)] = \mathbb{E} \left[ \sum_{k=0}^{K} c_k(t) \phi_k(\xi) \right] = c_0(t) \mathbb{E}[\phi_0(\xi)] + \mathbb{E} \left[ \sum_{k=1}^{K} c_k(t) \phi_k(\xi) \right] = c_0(t),
\]

(3.17)

\[
\text{Var}[U^K(t, \xi)] = \mathbb{E} \left[ (U^K(t, \xi) - E[U^K(t, \xi)])^2 \right] \]

\[
= \mathbb{E} \left[ \left( \sum_{k=0}^{K} c_k(t) \phi_k(\xi) - c_0(t) \right)^2 \right] \]

\[
= \mathbb{E} \left[ \left( \sum_{k=1}^{K} c_k(t) \phi_k(\xi) \right)^2 \right] = \sum_{k=1}^{K} c_k^2(t) \gamma_k.
\]

(3.18)

The multi-dimensional case results analogously to the univariate one. Denote by $k = (k_1, \ldots, k_d) \in \mathbb{N}_0^d$ a d-dimensional *multi-index* with magnitude $|k| = \sum_{i=1}^{d} k_i$, satisfying the ordering $i \leq j \Leftrightarrow i_i \leq j_i$, $i = 1, \ldots, d$. Using the assumption of mutually independent inputs, it follows that the d-variate basis functions of total degree less or equal to $K$ are defined as:

\[
\Psi_k = \phi_{k_1} \ldots \phi_{k_d}, 0 \leq |k| \leq K.
\]

(3.19)

The orthogonality condition satisfied by the multi-variate polynomials reads as:

\[
\mathbb{E}[\Psi_i(\xi) \Psi_j(\xi)] = \delta_{ij} \gamma_i,
\]

(3.20)

where $\delta_{ij} = \delta_{i_1 j_1} \ldots \delta_{i_d j_d}$ and $\gamma_i = \gamma_{i_1} \ldots \gamma_{i_d}$. Thus, in the multi-dimensional setting, the $(K + 1)^{th}$-order series approximation of a random process $U^K(t, \xi)$ is:

\[
U^K(t, \xi) = \sum_{|k|=0}^{K} c_k(t) \Psi_k(\xi).
\]

(3.21)

The alternative relation based on a single index was outlined in Equation 3.15. Formulas 3.17 and 3.18 hold for the multi-variate case ([47]).

In the following, the employed orthogonal polynomials are described. The authors from [61] presented a correspondence between the polynomials from the Askey scheme and the commonly used pdfs. When the used random variables are of Gaussian nature, the corresponding polynomials are the ("probabilist") Hermite polynomials ($H_0 = 1$, $H_1 = x$, $H_2 = x^2 - 1$, $H_3 = x^3 - 3$, etc.), having the normalization constant $\gamma_i = i!$. On the other hand, when the underlying random variables are uniformly distributed, the corresponding polynomials are the Legendre polynomials ($L_0 = 1$, $L_1 = x$, $L_2 = \frac{1}{2}(3x^2 - 1)$, $L_3 = \frac{1}{2}(5x^3 - 3x)$, etc.), with the normalization constant $\gamma_i = \frac{1}{2(2n+1)}$, $n$ being their corresponding degree. In the next section, the computation of the expansion coefficients $c_k$ is presented.
3.3 Spectral Methods for Uncertainty Assessment

3.3.2 Pseudo-spectral Approach

The pseudo-spectral approach, also termed discrete projections, or non-intrusive spectral expansion, is used to compute the coefficients from Equation 3.15 ([47, 59]). For simplicity, the method is first outlined for the univariate case. Note that for everything that follows, the terms pseudo-spectral approach and discrete projections are used interchangeably.

The one dimensional spectral expansion reads as:

\[ U^K(t, \xi) = \sum_{k=0}^{K} c_k(t) \phi_k(\xi) \).

(3.22)

Multiplying on the right hand side of the equation with an arbitrary polynomial \( \phi_i \), taking the expectation of both sides (c.f. Equation 3.3), and employing the orthogonality of the polynomials, it follows that:

\[ c_k = \frac{1}{\gamma_k} \int_{\Gamma} U^K(t, \xi) \phi_k(\xi) \rho_\xi d\xi = \frac{1}{\gamma_k} \mathbb{E}[U^K(t, \xi) \phi_k(\xi)], k = 0, \ldots, K. \]

(3.23)

As in most applications one only has access to the numerical version of \( U^K(t, \xi) \), the integral from Equation 3.23 is computed numerically via a quadrature or cubature rule ([59]). The optimal methodology for the one dimensional setting is based on Gaussian quadrature. This choice is motivated by the fact that for an input of size \( N \), Gaussian quadrature is exact for polynomials up to degree \( 2N-1 \). Furthermore, the Hermite and Legendre polynomials are orthogonal with respect to the normal and uniform probability measures, which represents the weight function in the above integral ([59, 61]). It follows that the coefficients in Equation 3.23 can be numerically approximated as:

\[ c_k(t) = \frac{1}{\gamma_k} \sum_{i=0}^{N-1} U^K(t, \zeta_i) \phi_k(\zeta_i) \omega_i, k = 0, \ldots, K, \]

(3.24)

where \( \{\zeta_i, \omega_i\}_{i=0}^{N-1} \) are the nodes and the weights of the quadrature rule. For a Gaussian quadrature rule of order \( N \), the nodes are the roots of the corresponding orthogonal polynomial of degree \( N+1 \), while the weights are the Lagrange interpolation polynomials evaluated at these nodes. When Hermite polynomials are employed, the rule is termed Gauss-Hermite quadrature, whereas for Legendre polynomials, Gauss-Legendre. For further reading about Gaussian quadrature, please refer to [43]. Note that once the set of nodes and weights is available, the computation of the coefficient becomes a deterministic problem.

Because normally distributed inputs were used in the majority of simulations carried out via the pseudo-spectral approach, in the following the computation of the coefficients is outlined only for this case, with the other resulting analogously.

Essentially all tables that specify the points for one-dimensional Gauss-Hermite quadrature employ the weight function \( \rho = e^{-x^2} \), which is used for the “physicist” Hermite polynomials ([47]). In order to use these nodes when the weight function is \( \rho_n = \frac{1}{\sqrt{4\pi}} e^{-\frac{x^2}{2}} \), a scaling of the given nodes and weights is required, which is achieved via the following substitution:

\[ \int_{\mathbb{R}} f(x) \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx \quad \Rightarrow \quad \int_{\mathbb{R}} f(\sqrt{2}y) e^{-y^2} dy. \]

(3.25)
3 Forward Propagation of Uncertainty

It follows that \( \{ \zeta_i, \omega_i \}_{i=0}^{N-1} \) is transformed to \( \{ \sqrt{2}\zeta_i, \omega_i \}_{i=0}^{N-1} \). In this way, using the scaled nodes and weights together with the normal linear transformation (c.f. Equation 3.9), when the uncertain input is modelled as a general normal random variable with mean \( \mu \) and standard deviation \( \sigma \), the coefficients of the spectral expansion are computed as:

\[
c_k(t) = \frac{1}{k!\sqrt{\pi}} \sum_{i=0}^{N-1} U^K(t, \sqrt{2\sigma\zeta_i} + \mu)\phi_k(\sqrt{2\zeta_i})\omega_i, \quad k = 0, \ldots, K. \tag{3.26}
\]

It is important to note that in both Gauss-Hermite and Gauss-Legendre quadratures, the arguments of the numerical solver need to be positive for the problem to make sense from a physical point of view. For Gauss-Hermite quadrature this reads as:

\[
\sqrt{2}\sigma\zeta_i + \mu \geq 0 \Leftrightarrow \frac{\mu}{\sigma} \geq -\sqrt{2}\zeta_i. \tag{3.27}
\]

Considering that the Gaussian quadrature employs both negative and positive nodes([43]), the ratio of the mean and standard deviation should be chosen with respect to the smallest node of the employed scheme.

The two dimensional case is outlined next, with the arbitrary-dimensional case presented afterwards. In this setting, the computation of the coefficients reads as:

\[
c_k = \frac{1}{\gamma_k} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} U^K(t, \mu_1 + \sigma_1 \xi_1, \mu_2 + \sigma_2 \xi_2)\phi_{k_1}(\xi_1)\phi_{k_2}(\xi_2)\rho_n(\xi_1)\rho_n(\xi_2)d\xi_1d\xi_2, \tag{3.28}
\]

where \( 0 \leq k \leq K - 1, \quad K = \frac{p(p+1)}{2} \), \( p \) is the number of interacting terms, \( \xi_1, \xi_2 \sim N(0, 1) \), \( \rho_n \) is the pdf for a standard normal random variable, \( \mu_1, \mu_2, \sigma_1, \sigma_2 \) are the means and the standard deviations of the inputs, respectively, and \( \gamma_k, \gamma_k^2 \) are the normalization constants for each dimension. Applying Fubini’s theorem ([43]) and considering un-normalized coefficients for simplicity, it follows that:

\[
c_k \propto \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} U^K(t, \mu_1 + \sigma_1 \xi_1, \mu_2 + \sigma_2 \xi_2)\phi_{k_1}(\xi_1)\phi_{k_2}(\xi_2)\rho_n(\xi_1)\rho_n(\xi_2)d\xi_1d\xi_2
\]

\[
= \int_{-\infty}^{\infty} \left( \int_{-\infty}^{\infty} U^K(t, \mu_1 + \sigma_1 \xi_1, \mu_2 + \sigma_2 \xi_2)\phi_{k_1}(\xi_1)\phi_{k_2}(\xi_2)\rho_n(\xi_1)d\xi_1 \right) d\xi_2 \tag{3.29}
\]

\[
= \int_{-\infty}^{\infty} \left( \int_{-\infty}^{\infty} U^K(t, \mu_1 + \sigma_1 \xi_1, \mu_2 + \sigma_2 \xi_2)\phi_{k_1}(\xi_1)\rho_n(\xi_1)d\xi_1 \right) \phi_{k_2}(\xi_2)d\xi_2.
\]

Thus, after integrating in the direction of \( \xi_1 \), the resulting functional is integrated in the other direction. Numerically, this translates as:

\[
c_k \propto \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} U^K(t, \sqrt{2}\sigma_1 \zeta_i + \mu_1, \sqrt{2}\sigma_2 \zeta_j + \mu_2)\phi_i(\sqrt{2}\zeta_i)\phi_j(\sqrt{2}\zeta_j)\omega_i\omega_j, \quad k = 0, \ldots, K, \tag{3.30}
\]

which is termed the tensor product or full grid based approach ([47, 61, 59]). In the current work, the full grid approach was implemented only for dimension two, when testing the first scenario (c.f. Section 5.1).

Denoting by \( I^* N(f) \) the one-dimensional quadrature rule of degree \( N \), in direction \( i \), applied to an integrable function \( f \), the computation of the coefficients for a \( d \)-dimensional
3.3 Spectral Methods for Uncertainty Assessment

case generalizes as:

\[ c_k \propto \left( I_{K1}^1 \times I_{K2}^2 \times \ldots \times I_{Kd}^d \right) (U^K(t, \xi) \phi_{k_1}(\xi_1) \phi_{k_2}(\xi_2) \ldots \phi_{k_d}(\xi_d) \rho_{n}(\xi_1) \rho_{n}(\xi_2) \ldots \rho_{n}(\xi_d)), \]

(3.31)

where \( \times \) denotes the tensor product \([43]\) and \( 0 \leq k \leq K, K = \frac{(p+0)!}{q!}. \) This implies that the necessary number of points for evaluating a coefficient in \( d \)-dimensions is in \( \mathcal{O}(N^d) \), which means an exponential growth in the number of points, thus, in the number of evaluations of the numerical solver. When a single evaluation of the solver is computationally expensive, even for a relatively small dimensionality, e.g. five, the full grid approach is infeasible from a computational point of view. This bears the name of the curse of dimensionality \([34, 39, 47, 53, 59]\) and several techniques have been developed to overcome it, up to a certain extent \([15, 34]\). In the current work, a hybrid sparse grids-one dimensional quadrature based approach was employed, detailed in Section 3.4.

3.3.3 Convergence and Properties

When the numerical solution exhibits smooth dependence on the inputs, the necessary truncation degree for obtaining an accurate approximation is typically low, leading to a low number of needed calls of the employed numerical solver \([61]\). After the coefficients were calculated, the computation of statistical properties is immediate (c.f. Equation 3.17 and Equation 3.18). In addition, as it is detailed in Section 3.6, the coefficients can be used to compute the sensitivity coefficients as well. Furthermore, the pdf of a QoI can be easily approximated using the spectral approximation 3.13 (more details in Section 3.5).

Even though the spectral expansion represents a surrogate model, the entire post-processing depends only on its coefficients, whose computation relies on the availability of the deterministic solver. Similar to MCS, this method is non-intrusive and once a code becomes available, it can be easily re-used.

The major contribution to overall computational cost is due to the calls of the numerical solver of the system. However, after all solutions are obtained, they are re-usable \([59, 60, 61]\). Hence, if the numerical solution is approximated via a rule of order \( N \), the computational cost is in \( \mathcal{O}(NC(1)) \), where \( C(1) \) is the cost of a single evaluation. In addition, the calls of the numerical solver are independent, making the method embarrassingly parallel and well suited for HPC.

According to the authors in \([39]\), the convergence rate is exponential and depends on the regularity of the solver. It reads as:

\[ \|U(t, \xi) - U^K(t, \xi)\|_{L^2(\Gamma)} \in \mathcal{O}(K^{-\alpha}), \]

(3.32)

where \( \alpha > 0 \) depends on the smoothness of \( U \). Note that even though in \([61]\), the authors proved the exponential convergence only for pdfs with associated polynomials from the Askey scheme, exponential convergence can be achieved for arbitrary distributions as well, via a technique pioneered in \([55]\), based on Gram-Schmidt orthogonalization. Even more, the spectral expansion method can be employed when the inputs are dependent random variables (see \([17]\) for more details).

In the following, a characterization of the numerical error of discrete projections is presented. Similarly to MCS, this method also inherits the numerical error of the solver. Moreover, the degree of truncation introduces an error reflected in the accuracy of the
approximation. Lastly, the computation of the integral via numerical quadrature introduces another error component. Denoting a generic numerical error with \( \epsilon \), by assigning appropriate indices, it follows that the overall numerical error can be outlined as:

\[
\epsilon_{DP} = \epsilon_N + \epsilon_Q + \epsilon_K,
\]

(3.33)

where \( N \) denotes the numerical solver, \( Q \) the quadrature and \( K \) the truncation order. Furthermore, when sparse grids interpolation is additionally used, it introduces another error component \( \epsilon_I \).

### 3.4 Discrete Projections using Sparse grids

As it was presented in Section 3.3.2, multi-dimensional UQ problems suffer from the curse of dimensions. As an example, consider that the coefficients of five dimensional spectral expansion approximation are to be computed using eight nodes and weights corresponding to a Gaussian-based quadrature rule. This results in 32768 number of points, thus, 32768 evaluations of the FSI solvers, which, given the underlying complexity of the solver, is an astronomical number. To overcome the curse of dimensionality, a hybrid sparse grids interpolation-quadrature approach was employed in the current work. In the following subsection, the general theory of the interpolation on sparse grids is outlined, compared to interpolation on a full grid. For both approaches, the computational cost and the resulting \( L_2 \) error are presented. After the theory of sparse grids interpolation is described, Section 3.4.2 presents the strategy used to compute the coefficients of the spectral expansion in high-dimensional settings. Finally, the employed sparse grid toolbox is described in the last subsection.

#### 3.4.1 Interpolation on Sparse Grids

The sparse grid idea was originally developed for the solution of partial differential equations ([25, 63]). However, it can be traced back to 1963, when the mathematician Smolyak ([48]) employed it in the context of numerical integration. The sparse grid method is a special discretization technique, suitable for overcoming the curse of dimensionality, to some extent, employing a hierarchical basis decomposition and a sparse tensor product construction ([10, 25, 63]). Due to the recent advancements in computing, high dimensionality is an intrinsic aspect of most problems. To overcome it, sparse grids were successfully applied to a wide variety of problems, ranging from integral equations ([27]), interpolation and approximation ([26]), to stochastic differential equations ([46]). Moreover, besides employing a hierarchical basis, a sparse grid representation can also be obtained by employing the combination technique (see [28] for more details). As sparse grids represent a complex domain on its own, the focus of this work is only on interpolation on sparse grids. For an extended overview of sparse grids, the reader is referred to [10]. The terminology adopted for the rest of this section is similar to the one in [10, 25, 42, 63].

In what follows, the working domain is denoted by \( \Omega = [0, 1]^d \), where \( d \) is the dimensionality. Note that a generalized finite domain \([a, b]\) can be obtained via a rescaling. Moreover, let \( l = (l_1, \ldots, l_d) \in \mathbb{N}^d \) denote the grid level, whereas \( i = (i_1, \ldots, i_d) \in \mathbb{N}^d \) denotes the spatial position. A full grid discretization of \( \Omega \) is denoted by \( \Omega_l \), consisting of points
3.4 Discrete Projections using Sparse grids

$x_{l,i} = (x_{l,i_1}, \ldots, x_{l,i_d})$, where $x_{l,i_t} = i_t h_{l_t} = i_t 2^{-l_t}$ and $i_t \in \{0, 1, \ldots, 2^{l_t}\}$. Considering a d-dimensional function $f : \Omega \to \mathbb{R}$, under the assumption $f|_{\partial \Omega} = 0$, a full grid interpolant $u$ of $f$ has the form:

$$f(x) = u(x) = \sum_i \alpha_i \varphi_i x,$$

(3.34)

with suitable basis functions $\varphi_i$ and coefficients $\alpha_i$ ([10]). In the next step, a hierarchical decomposition of the underlying approximation space is considered. The respective basis comprises hat functions, whose mother function is of the form:

$$\varphi(x) = \begin{cases} 1 - |x|, & \text{if } x \in [-1,1] \\ 0, & \text{else}. \end{cases}$$

(3.35)

The basis functions $\varphi_{l,i}$ with support $[x_{l,i} - h_l, x_{l,i} + h_l]$ are obtained via dilation and translation of the mother function, i.e. :

$$\varphi_{l,i} = \varphi \left( \frac{x - ih_l}{h_l} \right).$$

(3.36)

An example of one-dimensional hierarchical functions is depicted in Figure 3.1.

![One dimensional hierarchical space with three levels](source [21])

In the next step, let $W_l = \text{span}\{\varphi_{l,i} : i \in I_l\}$ to define the mutually disjoint hierarchical increment spaces, employing the index set $I_l = \{i \in \mathbb{N} : 1 \leq i \leq 2^l - 1, i \text{ odd}\}$. It follows that the corresponding function space denoted by $V_l$ is: $V_l = \bigotimes_{k \leq l} W_k$. Hence, the one-dimensional interpolant can be written as $u(x) = \sum_{k=1}^l \sum_{i \in I_k} \alpha_{k,i} \varphi_{k,i}(x)$. The $d$-variate extension is realized by a tensor product construction, i.e. :

$$\varphi_{l,i} = \prod_{j=1}^d \varphi_{l,i_j}(x_j),$$

(3.37)

with the index set $I_l = \{i \in \mathbb{N}^d : 1 \leq i_j \leq 2^{l_j} - 1, i_j \text{ odd}, 1 \leq j \leq d\}$ and hierarchical increments $W_l = \text{span}\{\varphi_{l,i} : i \in I_l\}$ ([10]). Defining $|l|_\infty = \max_{1 \leq i \leq d} |l_i|$, the function space becomes $V_n = \bigotimes_{|l|_\infty \leq n} W_l$, with the corresponding multi-dimensional interpolation function of the form:

$$u(x) = \sum_{|l|_\infty \leq n} \sum_{i \in I_l} \alpha_{l,i} \varphi_{l,i}(x),$$

(3.38)
where $n$ is the level. From the above formula it can be seen that the interpolant employs $(2^n - 1)^d$ grid points. Moreover, for a sufficiently smooth $f$ and $u$, the $L_2$ error reads as:

$$\|f(x) - u(x)\|_{L_2} \in \mathcal{O}(h_n^2),$$

(3.39)

but at a cost of $\mathcal{O}(h^{-d}_n) = \mathcal{O}(2^{nd})$ function evaluations. This exponential growth of the needed number of points represents the curse of dimensionality for the interpolation problem ([10]).

The question that follows is if one can construct discrete approximation spaces that are “better”, in the sense that the same number of employed grid points leads to a higher accuracy. Should the second mixed derivatives be bounded, the answer is yes ([10]). The functions obeying this condition belong to a Sobolev space $H^{\min}_2(\Omega) = \{f : \Omega \to \mathbb{R} : D^l f \in L^2(\Omega), |l|_\infty \leq 2, f|_{\partial \Omega} = 0\}$, where $D^l f = \frac{\partial^{d_1}}{\partial x_1^{d_1}} \cdots \frac{\partial^{d_d}}{\partial x_d^{d_d}} f$ and $|l|_1 = \sum_{j=1}^d l_j$ ([63]). In this setting, it can be proven that the hierarchical coefficients have an exponential decay, i.e. $|\alpha_{l,i}| = \mathcal{O}(2^{-2|l|_1})$ ([10]).

In words, the above formulas translate as leaving out the subspaces from the full grid that contribute only little to the overall interpolant. Mathematically, this represents an optimization problem (knapsack problem) ([10]), consisting of finding the optimum choice of subspaces such that the above conditions are fulfilled. This leads to a sparse grids space, which at level $n$ reads as: $V^S_{0,n} = \bigotimes_{|l|_1 \leq n + d - 1} W_l$. As an example, figure 3.2 depicts a two-dimensional sparse grid. It follows that the sparse grid interpolant has the form:

$$u(x) = \sum_{|l|_1 \leq n + d - 1} \sum_{i \in \Pi_l} \alpha_{l,i} \varphi_{l,i}(x),$$

(3.40)

where the number of employed grid points is $\mathcal{O}(h_n^{-1}(\log(h_n^{-1}))^{d-1}) \ll \mathcal{O}(h_n^{-d})$, with the corresponding $L_2$ error ([10]):

$$\|f(x) - u(x)\|_{L_2} \in \mathcal{O}(h_n^2(\log(h_n^{-1}))^{d-1}).$$

(3.41)
The coefficients are often termed *hierarchical surpluses* and they correct the interpolant of level \( l - 1 \) at \( x_{l,i} \) to the actual value of \( f(x_{l,i}) \) ([42]). Under the assumption \( f|_{\partial \Omega} = 0 \), the coefficients are the evaluation of \( f \) at the grid points.

Besides piecewise linear functions, another type of basis functions with local support employed in the current work is *piecewise polynomial functions*, depicted in Figure 3.3. For further reading, please refer to [42].

![Figure 3.3: One-dimensional piecewise polynomial basis functions of (maximum) degree \( p = 4 \) for up to level 3 (source [42])](image)

To allow non-zero values on the boundary of the domain, extra grid points located *directly on \( \partial \Omega \)* are considered together with additional basis functions that are non-zero at the boundary points. A two dimensional example is depicted in Figure 3.4. However, it should be noted that when a sparse grid with boundaries is employed, the number of grid points at the boundary is very large, comprising almost all points on \( \partial \Omega \). As an example, a level 4 linear boundary grid for a five dimensional problem comprises 2882 points. That is why, in the current work, a hybrid approach was employed that was based on using two sparse grids. One grid was used for the inner domain, i.e. without points on the boundary, of level 3 or 4, and the other for the boundary, i.e. without grid points in the interior, of level 0 or 1.

![Figure 3.4: One-dimensional basis functions with level 1 being extended with two extra level 0 basis functions at the boundary (left) and the corresponding grid (right) (source [42])](image)
3.4.2 Pseudo-spectral Approach using Sparse Grids

In this section, the pseudo-spectral approach using sparse grids is illustrated for the case when the input random variables are of Gaussian nature, which was the setting used throughout simulations. Moreover, the aforementioned approach was employed for UQ problems with stochastic dimensionality two and five.

Considering that interpolation on sparse grids lives on $[0, 1]^d$, whereas UQ problems employing the normal distribution have the support $\mathbb{R}^d$, where $d$ is the dimension, a mapping $\Lambda : [0, 1]^d \rightarrow \mathbb{R}^d$ is required. Note that due to the normal linear transformation (c.f. Equation 3.9), the computation of the coefficients of the spectral expansion approximation depends only on standard normal random variables.

The construction of this mapping is carried out by employing one important property of the standard normal distribution. The visual representation of the distribution is depicted in Figure 3.5. For a given $x > 0$, it is known that:

$$P(-x \leq \xi \leq x) = \int_{-x}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt = 2\text{erf}(x),$$

where $\text{erf}(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} dt$ is the erf function ([31]). In words, the above formula states that with probability $2\text{erf}(x)$, $\xi$ assumes values between $-x$ and $x$. In particular, if $x = 4$, $P(-4 \leq \xi \leq 4) = 0.9998$, i.e. 99.98% of the information is between ±4 standard deviations ([31]). It follows that if the range of $\Lambda$ is $[-4, 4]^d$ instead of $\mathbb{R}^d$, only 0.02% of the information is lost. Finally, the transformation $\Lambda$ is defined as:

$$\Lambda : [0, 1]^d \rightarrow [-4, 4]^d, \Lambda_i(x_i) = -4 + 8x_i, i = 1, \ldots, d.$$  (3.43)

For simplicity, in what follows, the two dimensional case is illustrated, with the other one resulting analogously. Equation 3.28 states how to evaluate the coefficients of the spectral expansion for a two dimensional UQ problem. In that equation, except for the numerical solver, everything else is decoupled in the two spatial directions, due to the tensor product construction of the basis polynomials $\Phi_k$ and to the stochastic independence of the inputs. In addition, as presented in the previous section, the multi-dimensional sparse grid interpolation basis functions are also constructed via a tensor product (c.f. Equation 3.40). Combining these two properties, it results that if one constructs an interpolant of
3.4 Discrete Projections using Sparse grids

Consider that the two inputs have the mean \( \mu = (\mu_1, \mu_2) \) and standard deviation \( \sigma = (\sigma_1, \sigma_2) \). Moreover, consider a boundary sparse grid of level \( n_b \), having the grid level vector \( l_b = (l_b^{(1)}, l_b^{(2)}) \) and \( K_b \) grid points, and a boundary free sparse grid of level \( n_i \), having the grid level vector \( l_i = (l_i^{(1)}, l_i^{(2)}) \) and \( K_i \) grid points. Note that subscripts \( b \) and \( i \) are used to differentiate between the interior and boundary grids, respectively. Furthermore, denote a boundary grid point as \( [0, 1]^2 \ni x_{ib} = (x_{ib}^{(1)}, x_{ib}^{(2)}) \), \( i = 0, \ldots, K_b - 1 \), whereas \( [0, 1]^2 \ni x_{ii} = (x_{ii}^{(1)}, x_{ii}^{(2)}) \), \( i = 0, \ldots, K_i - 1 \) denotes an interior grid point. Finally, let \( [0, 1]^2 \ni x = (x_1, x_2) \) denote an arbitrary point in the unit square.

In the first step, the numerical solution is interpolated at the boundary. Denoting the resulted interpolant by \( \hat{U}^K_b \), it reads as (c.f. Equation 3.40):

\[
\hat{U}^K_b(t, x) = \sum_{|l_b| \leq n_b + 1, i_b \in I_{l_b}} \alpha_{l_b, i_b}(t) \varphi_{l_b, i_b}(x),
\]

where \( \alpha_{l_b, i_b}(t) = U^K(t, \mu + \sigma \Lambda(x_{ib})) \) are the evaluations of the numerical solution at the boundary.

The second step consists of interpolating the solution inside the domain. The interpolant, denoted by \( \hat{U}^K_i \), reads as:

\[
\hat{U}^K_i(t, x) = \sum_{|l_i| \leq n_i + 1, i \in I_{l_i}} \alpha_{l_i, i}(t) \varphi_{l_i, i}(x_1) \varphi_{l_i, i}(x_2),
\]

where \( \alpha_{l_i, i}(t) = U^K(t, \mu + \sigma \Lambda(x_{ii})) - U^K_b(t, x_{ib}) \) are the hierarchical surpluses, computed as the difference between the evaluation of the numerical solver at the interior points and the one at the boundary ([42]).

Finally, the interpolant \( \hat{U}^K \) of the numerical solution is computed as the sum of the two aforementioned approximations, i.e.:

\[
\hat{U}^K(t, x) = \hat{U}^K_b(t, x) + \hat{U}^K_i(t, x)
\]
follows that:

\[
c_k = \frac{\text{vol}}{\gamma_{k_1} \gamma_{k_2}} \int_{[0,1]} \int_{[0,1]} \tilde{U}^K(t, x) \phi_{k_1}(A_1(x_1)) \phi_{k_2}(A_2(x_2)) \rho_n(A_1(x_1)) \rho_n(A_2(x_2)) \, dx_1 dx_2
\]

\[
= \frac{\text{vol}}{\gamma_{k_1} \gamma_{k_2}} \int_{[0,1]} \int_{[0,1]} (\tilde{U}_b^K(t, x) + \tilde{U}_1^K(t, x)) A(x_1, x_2) \, dx_1 dx_2
\]

\[
= \frac{\text{vol}}{\gamma_{k_1} \gamma_{k_2}} \left( \sum_{|l_i| \leq n_1 + 1, i_1 \in I_{k_1}} \alpha_{l_i, i_1, i_2}(t) \varphi_{l_i, i_1}^{(1)}(x_1) \varphi_{l_2, i_2}^{(2)}(x_2) A(x_1, x_2) \, dx_1 dx_2 \right) + \sum_{|l_i| \leq n_1 + 1, i_1 \in I_{k_1}} \alpha_{l_i, i_1, i_2}(t) \varphi_{l_i, i_1}^{(1)}(x_1) \varphi_{l_i, i_2}^{(2)}(x_2) A(x_1, x_2) \, dx_1 dx_2 \right),
\]

(3.49)

where \(0 \leq k \leq K-1, K\) being the number of terms, and \(\text{vol} = (4 - (-4))^2 = 8^2\) is the \textit{volume} resulting from changing the integration area to \([0, 1]^2\). For simplicity, denote \(\gamma = \frac{\text{vol}}{\gamma_{k_1} \gamma_{k_2}}\).

Taking advantage of the spatial independence of the constituents of \(A(x_1, x_2)\) and of the basis functions of the sparse grid, by pulling the sum out at the outer most level, the above formula becomes:

\[
c_k = \gamma \left( \sum_{l_i, i_1} \alpha_{l_i, i_1}(t) \int_{[0,1]} \varphi_{l_i, i_1}^{(1)}(x_1) \phi_{k_1}(A_1(x_1)) \rho_n(A_1(x_1)) \, dx_1 \right) \int_{[0,1]} \varphi_{l_2, i_2}^{(2)}(x_2) \phi_{k_2}(A_2(x_2)) \rho_n(A_2(x_2)) \, dx_2
\]

\[
+ \sum_{l_i, i_1} \alpha_{l_i, i_1}(t) \int_{[0,1]} \varphi_{l_i, i_1}^{(1)}(x_1) \phi_{k_1}(A_1(x_1)) \rho_n(A_1(x_1)) \, dx_1 \int_{[0,1]} \varphi_{l_2, i_2}^{(2)}(x_2) \phi_{k_2}(A_2(x_2)) \rho_n(A_2(x_2)) \, dx_2
\]

\[
= \gamma \left( \sum_{l_i, i_1} \alpha_{l_i, i_1}(t) \int_{J_{l_1}} \int_{J_{l_2}} + \sum_{l_i, i_1} \alpha_{l_i, i_1}(t) \int_{J_{l_1}} \int_{J_{l_2}} \right)
\]

(3.50)

The one dimensional integrals \(J_{l_1}, J_{l_2}, J_{i_1}, J_{i_2}\) were evaluated via a degree \(N\) Gauss-Legendre quadrature rule. Because this rule lives on \([-1, 1]\), the Transformation 3.8 is employed from \([-1, 1]\) to \([0, 1]\). It follows that e.g. \(J_{i_1}\) was computed as:

\[
J_{i_1} = \int_{[0,1]} \varphi_{l_i, i_1}^{(1)}(x_1) \phi_{k_i}(A_1(x_1)) \rho_n(A_1(x_1)) \, dx_1
\]

\[
= \frac{1}{2} \int_{[-1,1]} \varphi_{l_i, i_1}^{(1)} \left( \frac{1}{2} x_1 + \frac{1}{2} \right) \phi_{k_i} \left( A_1 \left( \frac{1}{2} x_1 + \frac{1}{2} \right) \right) \rho_n \left( A_1 \left( \frac{1}{2} x_1 + \frac{1}{2} \right) \right) \, dx_1
\]

\[
\approx \frac{1}{2} \sum_{j=0}^{N-1} \varphi_{l_i, i_1}^{(1)} \left( \frac{1}{2} \zeta_j + \frac{1}{2} \right) \phi_{k_i} \left( A_1 \left( \frac{1}{2} \zeta_j + \frac{1}{2} \right) \right) \rho_n \left( A_1 \left( \frac{1}{2} \zeta_j + \frac{1}{2} \right) \right) \omega_j,
\]

(3.51)

where \(\{\zeta_i, \omega_i\}_{i=0}^{N-1}\) are the nodes and the weights of the quadrature rule. All others one dimensional integrals were computed analogously.
3.4.3 Sparse Grids Toolbox SG++

In the current work, the sparse grids specific functionality is provided by SG++\(^1\). The toolbox was originally developed at Technische Universität München, and currently, jointly at Universität Stuttgart and Technische Universität München. It is implemented in C++, offering a broad spectrum of sparse grids related functionality. The main reference is [42].

It is a software package allowing the employment of \textit{spatially adaptive sparse grids} in a straightforward and flexible manner, \textit{out-of-the box}, without the overhead required by implementing the sparse grids and corresponding algorithms. The application spectrum ranges from interpolation and quadrature, to classification, and many others. Its main two features are \textit{efficiency} and \textit{flexibility}. The first one is explained by the C++ implementation, which ensures run-time performance. Moreover, it \textit{supports modern shared and multi-core systems} to speed-up computations for some performance critical operations. Flexibility is outlined by a \textit{modular architecture}, and \textit{separation of data structure and algorithms}, with all employed data structures already encapsulated. Furthermore, it supports different types of basis functions, enabling the possibility to deal with a wide range of applications. Even more, the adaptive refinement criteria can be easily specified as functor objects. Some of the available modules are: \textit{base}, \textit{quadrature}, \textit{optimization}, \textit{pde} and \textit{finance}.

SG++ can be used from within C++, Python, Java, and Matlab. However, in the current work it was employed only in C++. Moreover, the \textit{base} module was used, which provides the needed interpolation functionality. Moreover, SG++ was used in the pre- and post-processing steps of a multi-dimensional UQ simulation. To offer a better understanding of how this was achieved, the steps involving calling functions from the base module were summarized in Algorithm 3.

\begin{algorithm}
\caption{SG++ functionality in a multi-dimensional UQ problem}
\begin{algorithmic}[1]
\Require \texttt{d}, \texttt{sg\_level} \Comment{problem’s dimensionality and employed hierarchical level}
\State generate a grid of level \texttt{sg\_level} for dimension \texttt{d}
\State scale all grid points from \([0,1]^d \to [-4,4]^d\)
\State compute the coefficients of the numerical solutions’ sparse grid interpolant
\State hierarchize the coefficients
\State use the interpolant’s basis functions for evaluating \texttt{d} one-dimensional integrals
\end{algorithmic}
\end{algorithm}

3.5 Probability Density Function Estimation

The post-processing of a UQ simulation typically consists of computing the statistics of the QoI, such as the mean and variance. Although these estimates usually provide useful information, they are only \textit{point estimates}. For obtaining an estimate over the entire support of the QoI, the solution is to approximate the corresponding pdf(s). For what follows, a pdf estimate is denoted by \(\hat{\rho}\).

The traditional way to approximate a pdf from data is to employ a \textit{brute-force approach} based on the histogram of the output, as depicted on the left side of Figure 3.6. The

\(^1\)see \url{http://www5.in.tum.de/SGpp/releases/index.html} and \url{http://sgpp.sparsegrids.org/} for more details
underlying idea consists of dividing the sample space into $N$ bins of equal width. From there, $\hat{\rho}$ is approximated as: $\hat{\rho}(x) = \frac{1}{N} \frac{\# x_i \text{ in the same bin as } x}{\text{bin width}}$. However, the choice of the bin locations together with their number influence the structure of the resulting density ([47]). Hence, this technique is difficult to implement for dimensions greater than one.

In the current work an improved approach was employed, namely Kernel Density Estimation (KDE). KDE is a non-parametric estimation technique that generalises easily in multi-dimensional settings. The KDE methodology is depicted on the right side of Figure 3.6. The difference with respect to the brute-force approach is that the estimation is carried out by employing a known kernel $K$, which needs to obey the following constraints: $\int K(u)du = 1$ and $K(-u) = K(u), \forall u \in \mathbb{R}$ ([31]). Examples of typically employed kernels are: Gaussian, tophat, Epanechnikov, exponential, linear, and cosine, whose visual representation is depicted in Figure 3.7. In the current work the Gaussian kernel was used, which is an exponential function of the form:

$$K(x, h) \propto e^{-\frac{x^2}{2h^2}} \quad (3.52)$$

The approximation of a one dimensional $\hat{\rho}$ via KDE reads as:

$$\hat{\rho}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right), \quad (3.53)$$

where $h$ is the smoothing parameter or bandwidth ([47]). The implementation of the KDE methodology was based on the functionality offered by Python’s scikit-learn framework\(^2\).

---

\(^2\)see [http://scikit-learn.org/stable/](http://scikit-learn.org/stable/) for more details
In the current work, KDE was employed for one-dimensional UQ problems using the pseudo-spectral approach, focusing on inputs modelled as generalized normally distributed random variables \( \xi \sim \mathcal{N}(\mu, \sigma^2) \). Upon evaluating the coefficients of the spectral expansion via Equation 3.24, all necessary information required for performing the spectral expansion approximation of \( U^K(t, \xi) \) becomes available. In order to perform the KDE at a chosen \( t = \tilde{t} \), the next step consists of generating a user defined number \( M \) of random variables \( \xi_i, i = 0, \ldots, M - 1 \), which are input in \( U^K(\tilde{t}, \xi) \), resulting in a vector \( \tilde{U^K} \) of length \( M \). Next, this vector is input in the employed KDE function that outputs the support and the range of the pdf estimate. Furthermore, each pdf estimation is output together with the histogram of the data. All the aforementioned steps were summarized in Algorithm 4. Note that the implementation from this work provides the option to employ KDE at all time steps, resulting in an animation comprising all estimated pdfs.

### Algorithm 4 KDE based pdf estimation for 1D discrete projection based UQ Simulations

| Require: \( M \) | \( \triangleright \) number of random variables \( \xi_i \) that are to be generated |
| Require: \( \mu, \sigma \) | \( \triangleright \) mean and standard deviation of the stochastic input |
| Require: \( \tilde{t} \) | \( \triangleright \) time-step when the pdf if estimated |

1: calculate the coefficients of the spectral expansion using Gaussian quadrature
2: generate \( \xi_i \sim \mathcal{N}(\mu, \sigma^2), i = 0, \ldots, M - 1 \)
3: for \( i \leftarrow 0 \) to \( M - 1 \) do
4: get \( U^K_i = \sum_{k=0}^{K} c_k(\tilde{t}) \phi_k(\xi_i) \) \( \triangleright \) QoI realization
5: append \( U^K_i \) to \( \tilde{U^K} \)
6: end for
7: input \( \tilde{U^K} \) in the employed KDE function

Ensure: pdf estimation at \( t = \tilde{t} \)

### 3.6 Variance Based Sensitivity Analysis

This section describes the variance based sensitivity analysis (SA) methodology, which was used in the post-processing step. It is not aimed to provide an exhaustive description of the subject, being only focused on the aspects relevant for the current work. For further reading, please refer to [44] and [45].

Broadly speaking, the aim of SA is to determine how the output uncertainty of a system can be apportioned to different sources of uncertainty in its inputs. SA methods are usually classified in two classes: local SA, which outlines the local impact of input parameters, by evaluating the gradient of the response with respect to its inputs, around a prescribed nominal value, and global SA, which aims to quantify the output uncertainty due to the uncertainty in the input parameters, taken singularly or in combination with others. Moreover, global sensitivity analysis (GSA) methods can be divided into two groups: regression-based methods and variance-based methods. In this project, variance-based GSA was employed. Variance based methods employ the decomposition of the variance of the output in a sum of contributions of each input variable and their combination ([49]). This technique is termed ANOVA, which stands for ANalysis Of VAriance.
In the current work, the aim of SA is to provide a detailed understanding of the variance of the QoI resulted after each simulation. As discrete projections represented the core UQ methodology, SA was carried out only when this method was employed. Furthermore, SA was performed via computing Sobol’s indices, taking advantage of the results described in [1] and [49] that outline a technique to compute Sobol’s indices using directly the coefficients of the spectral expansion approximation.

To illustrate the SA methodology, consider for simplicity a d-dimensional model \( U(x) \) defined on \([0, 1]^d\). The variance decomposition or the Sobol-Hoeffding decomposition ([49]) reads as:

\[
U(x) = U_0 + \sum_{i=1}^{d} U_i(x_i) + \sum_{1 \leq i < j \leq d} U_{ij}(x_i, x_j) + \ldots U_{12..d}(x),
\]

(3.54)

where \( U_0 \) is the mean of the model, \( U_i(x_i) \) are univariate functions, \( U_{ij}(x_i, x_j) \) are bivariate functions, etc. If the orthogonality of the terms is imposed on \([0, 1]^d\), the above functional decomposition becomes unique ([49]). The terms are computed recurrently using the orthogonality constraint as:

\[
U_i(x_i) = \int_{[0, 1]^{d-1}} U(x) dx_{\sim i} - U_0,
\]

\[
U_{ij}(x_i, x_j) = \int_{[0, 1]^{d-2}} U(x) dx_{\sim ij} - U_i(x_i) - U_j(x_j) - U_0,
\]

(3.55)

where \( \int_{[0, 1]^{d-1}} \) means that the integration is performed over all variables, except the \( i^{th} \). Going further and assuming that \( x_i \sim U(0, 1), i = 1, \ldots, d \), from the orthogonality of the decomposition it follows that the total variance \( D \) is:

\[
D = \sum_{\{i_1, \ldots, i_s\} \subset \{1, \ldots, d\}} \mathbb{E}[U_{i_1i_2\ldots i_s}^2(X_{i_1} \ldots X_{i_s})].
\]

(3.56)

Denoting \( D_{i_1\ldots i_s} = \int_{[0, 1]^s} U_{i_1\ldots i_s}^2(x_{i_1} \ldots x_{i_s}) dx_{i_1} \ldots dx_{i_s} \), it follows that:

\[
D = \sum_{i=1}^{d} D_i + \sum_{1 \leq i < j \leq d} D_{ij} + D_{12..d}.
\]

(3.57)

The Sobol’s indices are obtained by normalizing the partial variances in the above formula ([44, 49]), i.e.:

\[
S_{i_1\ldots i_s} = \frac{D_{i_1\ldots i_s}}{D}.
\]

(3.58)

If in addition one wants to quantify the total contribution of an input parameter to the overall variance, the individual index together with indices associated to combinations involving that parameter must be added up. This results in a total Sobol’s index, computed as:

\[
S_i^T = \sum_{i \subset \{i_1, \ldots, i_s\}} S_{i_1, \ldots, i_s}.
\]

(3.59)
Note that the all the above formulas hold true for generic spaces that are orthogonal with respect to the pdf used to model the input(s). Hence, the formulas hold true for both normal and uniform distributions.

In the current work, only local indices were computed, for the following two reasons. On one hand, the local indices quantify the contribution of each input parameter to the total variance. This emphasizes the parameter or parameters whose input uncertainty contributes the most to the overall resulted uncertainty. On the other hand, the local indices also outline the contribution to the total variance attributed to combinations of input parameters. This shows whether the output uncertainty consists of independent contributions of the uncertain inputs or not, providing a mean to assess whether a multi-dimensional UQ simulation outlines more information than the one or lower-dimensional counterparts. In what follows, Sobol’s indices computation using the coefficient of the spectral expansion approximation is presented.

As shown in Equation 3.18, the total variance is computed directly using the coefficients of the spectral expansion. It follows that in order to determine Sobol’s indices as outlined in Equation 3.58, the next step is to find a method to compute $D_{i_1...i_s}$ using a suitable subset of coefficients. To this extent, consider a generic interaction set, i.e. a set of multi-indices depending exactly on the subset of variables $i = \{i_1 \ldots i_s\}$ ([49]), defined as:

$$A_i = \{c \in \mathbb{N}^d : k \in i \iff c_k \neq 0\}, \cup_{i \subseteq \{1, \ldots , d\}} A_i = \mathbb{N}^d.$$  \hfill (3.60)

Replacing $U(x)$ with $U^K(t, \xi)$ and employing the uniqueness of the Sobol’s decomposition, it follows that:

$$U^K(t, \xi) = U_0(t) + \sum_{i \subseteq \{1, \ldots , d\}} U_i(\xi_i),$$  \hfill (3.61)

where $U_i(\xi_i) = \sum_{k \in A_i} c_k(t) \psi_k(\xi)$. From the above formula and Equation 3.57, it follows that the partial variances $D_i = D_{i_1...i_s}$ can be computed by summing up the square of the expansion coefficients from the corresponding interaction set, as:

$$D_i = \sum_{k \in A_i} c_k^2, \quad A_i = \{c \in \mathbb{N}^d : c_i > 0, c_j \neq i = 0\},$$  \hfill (3.62)

for the first order terms, and

$$D_i = \sum_{k \in A_i} c_k^2, \quad A_i = \{c \in \mathbb{N}^d : i \in i \iff c_i > 0\}$$  \hfill (3.63)

for the higher order contribution. Using the above derivations, Sobol’s indices are computed as:

$$S_i = \frac{D_i}{D}.$$  \hfill (3.64)

As a final remark, by taking into account the emphasized computation of statistics, pdf estimation via KDE, and the computation of the Sobol’s indices used for SA, the post-processing step when employing the pseudo-spectral approach is performed depending only on the coefficients of the spectral expansion, at virtually no additional computational cost.
3 Forward Propagation of Uncertainty
4 Implementation Details

The purpose of this chapter is to present the implemented UQ code and is divided into four sections. A general description is given in the first section, providing an overview of the implementation, of its functionality and properties, as well as its interaction with the application code. Moreover, a general work-flow diagram is outlined. The next three sections present a more detailed description, focusing on the implementation’s constitutive modules: pre-processing, described in Section 4.2, UQ simulation phase, described in Section 4.3 and at the end, the post-processing step, described in Section 4.4.

4.1 General Description

The UQ software consists of a main C++ implementation together with a set of external tools used for interacting with the application code. One of its main features is its non-intrusive interaction with the FSI software, by only needing access to the configuration files and the executable file of the multi-physics software, hence, treating an application code as a black box. Furthermore, its configuration is done via an external text file at runtime, facilitating e.g. the switch between UQ methods or probability distributions, without the need for recompiling the code. Another important feature is its division into three main, independent modules: pre-processing, simulation, and post-processing. This implies that the code is easily comprehensible and extensible. Last but not least, although it was tested for only two applications, it can be easily altered to work for any application having a similar architecture to the one considered in this work.

Explained in more details in the following sections, the UQ simulation work-flow is depicted in Figure 4.1. Before going any further, it is important to point that that the location of the UQ code is assumed to be:

/path/to/test_case/uq_code,

which implies that everything that interacts with the application code requires the corresponding relative path.

The configuration file is a text file named configuration.uq, located inside the folder uq_code/, obeying the following editing rules:

- lines that start with # are ignored
- configuration lines are of the form: name = value
- no line is allowed to be empty.

Hence, an example line is nsamples = 100. Moreover, it comprises three main parts, each responsible for a particular type of configuration:
Figure 4.1: Entire UQ simulation work-flow
1. communication with the application code configuration

In this part, the relative path to the external tools and specific application code files is set. As an example, the set-up for the relative path to the tool used to modify the value of the density of the fluid is: `insert_nastin_exec = ../tools/util/insert_params_nastin_dens`.

2. UQ problem configuration

In this segment, the user sets up UQ specific data, such as which UQ algorithm is to be used, the employed pdf used to model input uncertainties, the number of samples for MCS, the quadrature degree for discrete projections, etc. It is required that the algorithm and pdf are specified as boolean variables, with the following convention: 0 for Monte Carlo, 1 for discrete projections, and 0 for normal distribution and 1 for uniform distribution respectively. As an example, the set-up for normal distribution is `pdf = 0`. If this rule is not met, a corresponding message is printed on the console, stating that the underlying input configuration is not available.

3. stochastic problem set-up

In the final part, the numerical values for the two parameters characterizing the uncertain input(s) are specified; for the normal distribution, the parameters are the mean and the standard deviation, whereas for the uniform distribution, the left and right interval boundaries, respectively. As an example, if the fluid density is modelled as a normal random variable with mean 0.01 and standard deviation 0.001, the set-up is done as `rho_f_p1 = 0.01`, followed by `rho_f_p2 = 0.001`.

The C++ implementation of the UQ code consists of a sequence of classes realizing the functionality of the algorithms outlined in Chapter 3, together with their additional requirements. This means that for the MCS, classes used to elicit a user defined number of normal or uniform random variables are implemented, while for discrete projections, the additional functionality consists of classes that generate nodes and weights used for numerical quadrature. Furthermore, additional functions are implemented in a separate helper library, which could be used to parse the configuration file, create strings with user defined names, used to call external tools, etc.. An important observation is that the external tools, as well as the multi-physics software’s executable file are called via system calls\(^1\). Moreover, besides the C++ implementation, a sequence of Python, Perl, and bash scripts were designed for the interaction with the application code, mainly in the pre- and post-processing phases.

The end of the implementation process resulted in six projects. The first three were serial UQ implementations, consisting of one for a one-dimensional problem, another for a scenario with two stochastic dimensions, and the last for a five-dimensional stochastic problem. These codes are constructed as single, stand-alone applications, comprising all the three aforementioned modules. Although they encompass an entire UQ simulation work-flow, they proved to be of little use for the testing phase. The first reason is due to their serial nature. Even though the solvers have parallel capabilities (c.f. Sections 2.3 and 2.4), a serial UQ implementation still means that only one sample is executed at a time. This approach is not suited to cope with the high computational demand of a UQ simulation, which requires the results from several complex FSI runs. One solution would be to transform the serial UQ code into a parallel one. This leads to the second reason for which the

---

\(^1\)the `int system (const char* command)` function is available in the `cstdlib` C++ library
serial UQ codes were not well suited for the testing phase. Parallelization of the UQ code would imply that the FSI solvers would be run in parallel, each by a separate process, but as the application codes are already internally parallel together with the fact that a single application cannot contain two levels of parallelism. i.e. MPI initialization cannot be performed twice, the only solution would be to employ the serial versions of the application codes. Considering that for one simulation with an output of only a few seconds, even the parallel version of the benchmark scenario needs 24 hours of computing time on 16 cores, a serial version of the solvers is not feasible. What would therefore be desirable is a UQ code that is somehow parallel and uses the parallel versions of the FSI solvers as well. To achieve this, the UQ implementation was decoupled into three separate applications, one for each constitutive module. This offered the possibility to run the simulations in a pseudo-parallel fashion, by taking advantage of the fact that the runs are performed on clusters using a batch system. To put this in perspective, after the pre-processing step is executed and a user defined number of UQ points is generated, each associated with one simulation, a subset of the entire set of simulations are run simultaneously, each on individual nodes, letting the cluster’s batch system to administrate their run, depending on the degree of cluster’s occupancy. Finally, after all simulations are performed, the post-processing phase is performed, using the third application. All the aforementioned steps are explained further in the next three sections.

The use of C++ as the main development language offered multiple advantages, including the possibility to create an implementation that is not only HPC-suited, but also permits the usage of modern programming paradigms, such as object-oriented programming.

![Diagram of C++ UQ implementation](image)

Figure 4.2: C++ UQ implementation

The above diagram depicts the skeleton class diagram of the C++ implementation. As

---

2 either samples, for MCS, or quadrature/interpolation nodes, for discrete projections

3 in this work, the MAC Cluster was used for running the FSI benchmark related simulations, where the maximum number of simulations that can be run simultaneously is 10; more details in Section 5.2
it can be seen, three interfaces were designed. Gaussian quadrature specifies the necessary states and behaviours needed for specific instances of Gaussian-based quadrature, employed in the discrete projections method. Similarly, the Random number generator characterizes the functionality utilized by the two random number generators, employed in MCS. Finally, a generic UQ simulation class was defined, which outlines the modular behaviour of a UQ algorithm, comprising pre-processing, simulation, and post-processing. Moreover, the functionality of the helper class is aggregated to all random numbers and quadrature implementations. Note that for multi-dimensional scenarios, the implementation of discrete projections was coupled with the SG++ library.

As it was previously stated, the UQ implementation was designed to interact with the multi-physics code in a non-intrusive way. In order to achieve that, it needs access to the configuration files for the fluid and structure solvers, from Alya’s nastin and solidz modules. As an example, for the first employed scenario (c.f. Section 5.1), the configuration files are Flap.nsi.dat - for the fluid solver and Flap.sld.dat - for the structure solver. The part where the physical parameters are set looks like:

```plaintext
... PROPERTIES DENSITY= 0.01 VISCOSITY= 0.03 END_PROPERTIES ...
for the fluid solver and

... PROPERTIES DENSITY= 0.01 CONSTITUTIVE_MODEL ISOLIN 0.5e5 0.3 END_PROPERTIES ...
for the structure problem, where, the numbers from the CONSTITUTIVE_MODEL line represent the Young’s modulus and Poisson’s ratio, respectively. Note that all FSI test cases implemented with Alya employ two configuration files for the fluid and structure problem, respectively, of the form Name.nsi.dat and Name.sld.dat, where Name is a conventional name of the test case.

In the following sections, the pre-processing, simulation, and post-processing steps are explained in more details. Before the pre-processing step is performed, it is assumed that the configuration is already set-up in configuration.uq.

4.2 Pre-processing

MCS

The first step in MCS is to generate a user-defined number of random variables, depending on the stochastic dimensionality of the problem. This is realized using a pseudo-random number generator from the C++ standard library\(^4\), based on a Mersenne Twister 19937 gen-

\(^4\)available in `#include <random>`
erator and a seed using a random device. This choices provide an accurate way of generating random variables. This functionality is implemented in the classes NormalRandomVariable and UniformRandomVariable. Via calling an external bash script tool called create_data_point, a folder named simulation\_i, \(i = 0...\text{nsamples} - 1\) is created, which contains nastin\_i and solidz\_i, \(i = 0...\text{nsamples} - 1\) folders, Alya’s executable (Alya.x) and the preCICE configuration file, respectively. Initially, the nastin\_i and solidz\_i contain the configuration file for a deterministic run. Succeeding in the creation of the simulation folder, the replacement of the deterministic physical parameters considered to be uncertain with the corresponding random variable follow. This is realized by two external Perl tools, one for the fluid data and the other for the data of the structure, who edit the data files in place. All steps are summarized in the following algorithm:

**Algorithm 5 Pre-processing for arbitrary dimension MCS**

**Require:** dimension, nsamples values \(\triangleright\) defined in the configuration file

1: for \(i \leftarrow 0\) to dimension \(- 1\) do
2: call function that creates \(n\) random variables
3: end for
4: for \(i \leftarrow 0\) to \(nsamples\) \(- 1\) do
5: call create data point \(\triangleright\) create simulation\_i folder
6: call insert new data \(\triangleright\) replace the underlying physical parameter in simulation\_i
7: end for

Discrete projections: classical approach

The classical discrete projection approach was implement for dimensions one and two, using a full grid approach. The first pre-processing step is to generate a user-defined number q of nodes and weights, where q represents the quadrature degree, whose value is set up in the configuration file. In the one-dimensional case, the functionality corresponding to the two Gaussian-based quadrature rules is implemented in the classes named GaussHermiteQuadrature and GaussLegendreQuadrature. The nodes and weights are saved statically

**Algorithm 6 Pre-processing for classical discrete projections**

**Require:** dimension, quadrature degree \(\triangleright\) values defined in the configuration file

1: if dimension is 1 then
2: call function that generates quadrature degree nodes and weights
3: else if dimension is 2 then
4: for \(i \leftarrow 0\) to quadrature degree \(- 1\) do
5: call function that generates quadrature degree nodes and weights
6: end for
7: end if
8: for \(i \leftarrow 0\) to quadrature degree \(- 1\) do
9: call create data point \(\triangleright\) create simulation\_i folder
10: call insert new data \(\triangleright\) replace the underlying physical parameter in simulation\_i
11: end for
using \textit{STL vectors}$^5$ for every input degree$^6$ less than or equal to 10. Even more, they are used for the brute-force two dimensional UQ problem as well, as outlined in Equation 3.30. The following steps are similar to the MCS case, with the difference that instead of random samples, the deterministic parameters are replaced with the \textit{scaled quadrature nodes} (c.f. Section 3.3.2). For a one dimensional quadrature of size $q$, the degree of the two-dimensional scenario is $q^2$. In this case, instead of using one-dimensional vectors for storing the nodes and weights, two matrices of dimension $2 \times q^2$ were employed, one for storing the nodes, the other for storing the weights, whose rows contain the tensor product of the one dimensional nodes or weights, respectively. What follows is again similar to MCS. All steps were summarized in Algorithm 6.

\textbf{Discrete projections: Sparse-grids based approach}

The sparse grids based approach was employed for two and five dimensional UQ scenarios. The difference between this and the classical approach stems from the \textit{generation of the UQ points}. While in the classical approach they were \textit{quadrature nodes}, in this case they represent \textit{point coordinates on a regular sparse grid}. That is why, the first step is to generate a grid of dimension \textit{dim} and level \textit{sg\_level}. Afterwards, the dimensionality of the grid is computed, in order to determine how many \textit{simulation\_i} folders to create. The next step consists of \textit{scaling all grid points} from $[0,1]^d \rightarrow [a,b]^d$, where $a$ and $b$ are the boundaries of the employed computational domain, which in the current work is $[-4,4]$ (c.f. Section 3.4). In order to achieve the latter, access to the grid points has to be provided. To this extent, in SG++ a \textit{grid index} is used, with which the \textit{coordinates, level, or index} of all grid points is retrieved. Finally, after all points are scaled to the employed computational domain, the last step is to scale the stochastic inputs, which were modelled as generic random variables, to standard random variables (c.f. Equations 3.8 and 3.9). The pre-processing step ends with the creation of the \textit{simulation\_i} folders, realized in a similar way as for MCS and for the classic approach. All steps were summarized in Algorithm 7.

\begin{algorithm}[H]
\begin{algorithmic}[1]
\State \textbf{Require:} \textit{dim, sg\_level} \Comment{values defined in the configuration file}
\State create sparse grid of dimension \textit{dim} and level \textit{sg\_level}
\State get the dimensionality of the sparse grid and save it into \textit{sg\_dim}
\State create a grid index
\For{$i \leftarrow 0$ \text{to} \textit{sg\_dim}}
\State get the grid point $x_i$ from the $i^{th}$ coordinate
\State scale $x_i$ from $[0,1]^d \rightarrow [a,b]^d$ as $\hat{x}_i = a + (b - a)x_i$
\EndFor
\For{$i \leftarrow 0$ \text{to} \textit{sg\_dim}}
\State call create\_data\_point \Comment{create simulation\_i folder}
\State call insert\_new\_data \Comment{replace the underlying physical parameter in simulation\_i}
\EndFor
\end{algorithmic}
\caption{Pre-processing for discrete projections using sparse grids}
\end{algorithm}

$^5$defined in the \textit{vector library}

$^6$for the Gauss-Legendre case, degree 20 is implemented as a special case, not for pre-processing purposes, but for the post-processing step of the sparse-grids based approach
4 Implementation Details

4.3 Simulation

Simulations were run for all data created in the pre-processing step, i.e. for all simulation\(_i\), \(i = 0, \ldots, n_{\text{points}} - 1\) folders. The structure of these folders is as follows:

```
simulation\(_i\)
  | nassin\(_i\)
  |   | Fluid’s solver data
  | solidz\(_i\)
  |   | Structure’s solver data
  | Alya.x
  | precice-config.xml
```

In the config file, the folder used to exchange information between the fluid’s and structure’s solvers is `../`, relative to `nassin\(_i\)` and `solidz\(_i\)`. Hence, it is `simulation\(_i\)`.

Furthermore, for a generic application `sim`, the fluid’s and structure’s solvers are run as `/Alya.x sim`, from inside the corresponding data folders. It follows that, having the above directory structure, the \(i^{\text{th}}\) simulation is executed as described in the algorithm below:

```
Algorithm 8 Execution of UQ simulation \(i\)

Require: \(i\) \quad \triangleright\text{the current UQ simulation}

1: change directory to `/path/to/simulation\(_i\)/nassin\(_i\)` \quad \triangleright\text{fluid’s solver data}
2: `../Alya.x sim`
3: change directory to `/path/to/simulation\(_i\)/solidz\(_i\)` \quad \triangleright\text{structure’s solver data}
4: `../Alya.x sim`
```

4.4 Post-processing

From the three constitutive modules, post-processing is the most complex, mainly due to the fact that after all simulations are performed, the resulted data can be analysed from multiple perspectives, e.g. from a statistics, probabilistic, or sensitivity analysis point of view.

The underlying implementation provides a broad spectrum of post-processing functionality, reflected by a set of C++ methods together with a set of external Python based tools. The C++ code performs the post-processing step for the watch point of the structure, saved at each time step by preCICE (c.f. Section sec:General Description of the Coupling Library). The rest of the post-processing step is carried out using the aforementioned set of Python tools, some of which are called directly in the C++ implementation, while other are used as external applications. The visualization of the post-processing results was performed using both Paraview\(^7\) and Python’s matplotlib library\(^8\).

In what follows, the description of the post-processing is made separately for the two implemented UQ algorithms, with a focus on discrete projections, for which the most functionality was created. Before going any further, it is important to note the similarities in

---

\(^7\)see [http://www.paraview.org/](http://www.paraview.org/) for more details

\(^8\)see [http://matplotlib.org/](http://matplotlib.org/) for more details
the post-processing step for both algorithms. This is reflected in the C++ implementation, where the first task consists of gathering the output from all simulations in a common directory. Denoting a generic number of simulations with \(n\), data collection was carried out via an external bash script tool named `postprocessing_alya` which gathers all output binary files in a directory located at the same level as `simulation_{i}, i = 0 \ldots n - 1`, named `alya_output`. This directory contains \(n\) nastin and solidiz folders, where the former comprises the binary data for pressure, velocity, mesh displacement, and mesh velocity at each time step, whereas the latter contains structure’s displacement data at each time step. Afterwards, the next step was carried out via an external Python tool and consisted of gathering all watch point’s data files in a common folder, named `data_results/`, located in `test case/uq_code/` directory.

**MCS**

Once the data files of the watch point are available, the statistical evaluation of the chosen QoI is carried out for each time step, according to the formulas outlined in Algorithm 2. Once their computation is finished, they are saved in file called `Statistics.mc.txt`, located in a folder named `postprocessing_results/MCS/`, found at the same level as `data_results`. Even more, the statistics computations can be achieved for all output quantities of a simulation. In the first step, the binary files need to be transformed into Ensight files, via the `alya2pos` tool. Afterwards, two Python tools named `MCS_postprocessing_mean` and `MCS_postprocessing_var` are used to compute the mean and variance, respectively. Note that the above description holds true independently of the problem’s dimensionality. All steps were summarized in Algorithm 9.

**Algorithm 9** Post-processing step for MCS

Require: \(nsamples\) \(\triangleq\) defined in the configuration file

1: for \(i \leftarrow 0\) to \(nsamples - 1\) do
2: call `get_alya_output` \(\triangleright\) move data from `simulation_{i}/` to `alya_output/`
3: call `gather_data_mc` \(\triangleright\) copy data from `simulation_{i}/` to `data_results/MCS/`
4: compute statistics for the chosen quantities of interest from the watch point
5: end for
6: for \(i \leftarrow 0\) to \(nsamples - 1\) do
7: call `alya2pos` to transform the binary files of the output into Ensight files
8: end for
9: call `MCS_postprocessing_mean` and `MCS_postprocessing_var` to compute the statistics for all output quantities \(\triangleright\) e.g. pressure, velocity, etc

**Discrete Projections**

The post-processing step for discrete projections was implemented for one, two, and five dimensional UQ simulations for the first tested scenario, and for the five dimensional UQ setting for the second scenario. From a functional point of view, it is similar for both classical and sparse grids based approach. Moreover, similar to MCS, it comprises a set of C++ methods and external Python based tools.

In the C++ code, after the data of the watch point is available, the first step is to compute the coefficients of the spectral expansion. As it was described throughout Section 3.3, they
are quintessential for the entire post-processing step. Note that their computation differs for a one or multi-dimensional problem, as well as for classic or sparse grids based approach. After their computation, the coefficients are saved in a text file generically named `coeff.sc.txt`, which is further processed by both C++ and Python implementations. In the former, the next step is to evaluate the statistics, which are afterwards saved in a file called `Statistics.sc.txt`, located in `postprocessing_results/SCS/`, found at the same level as `data_results`. If the statistics evaluation is performed for every considered dimension, the functionality provided by the Python tools is dimension specific. In the one-dimensional setting, two tools were implemented for carrying out the pdf estimation via KDE, one for a specific user defined time-step, while the second estimates the pdf for all time steps, producing an animation. For two and five dimensions, the pdf estimation via KDE was not implemented, but additionally to the statistics estimation, a tool designed to perform the variance based sensitivity analysis was created.

The post-processing step for discrete projections comprises functionality used for the evaluation of the entire output of a simulation also, with the observation that it was implemented only for the dimensionality one and two in the case of the first tested scenario, for which the end goal was to obtain a comprehensive UQ analysis. For the second scenario, the analysis was performed only for the QoI from a specific location, i.e. the watch point of the structure. To this extent, after the output files are transformed in Ensight format, the statistics are evaluated for every considered dimension, the functionality provided by the Python tools is dimension specific. In the one-dimensional setting, two tools were implemented for carrying out the pdf estimation via KDE, one for a specific user defined time-step, while the second estimates the pdf for all time steps, producing an animation. For two and five dimensions, the pdf estimation via KDE was not implemented, but additionally to the statistics estimation, a tool designed to perform the variance based sensitivity analysis was created.

The post-processing step for discrete projections comprises functionality used for the evaluation of the entire output of a simulation also, with the observation that it was implemented only for the dimensionality one and two in the case of the first tested scenario, for which the end goal was to obtain a comprehensive UQ analysis. For the second scenario, the analysis was performed only for the QoI from a specific location, i.e. the watch point of the structure. To this extent, after the output files are transformed in Ensight format,
the spectral expansion coefficients were computed for each time step, by employing an external tool. Using the coefficients, the next step consists of the statistics evaluation.

In the case of the first scenario, pdf estimation and variance based sensitivity analysis were performed as well, for QoI corresponding to two coordinates from the fluid domain. Note that as the space discretization for the first tested scenario was realized via FEM, these coordinates represent coordinates of the two dimensional elements in the range 1, \ldots, 1373. To this extent, additional tools were implemented to gather the chosen QoI for the two fluid points. All steps were summarized in Algorithm 10.
4 Implementation Details
5 Test Scenarios and Results

The purpose of this chapter is to present the obtained UQ results. The simulation part employed two FSI scenarios, namely vertical flap and an FSI benchmark scenario. The description of these scenarios, together with the corresponding results is made in Sections 5.1 and 5.2, respectively. Section 5.1 begins with the physical description, having the purpose to familiarize the reader with the characteristics of the vertical flap scenario. It is followed by the description of deterministic simulations. Finally, the UQ results one (Section 5.1.2), two (Section 5.1.3), and five (Section 5.1.4) UQ settings are presented. The section describing the second scenario outlines its physical description, comprising domain related definitions, initial and boundary conditions, and the set-up of the physical parameters. It continues with the presentation of the results of a deterministic simulation, finalizing with the results obtained for the five-dimensional UQ simulation. For everything that follows, to differentiate between fluid and structure, subscripts $f$ and $s$ were used.

5.1 Vertical Flap

The first simulated scenario was a vertical flap, depicted in Figure 5.1. It consists of a two dimensional fluid channel flow, together with a vertical elastic structure, which deforms while the fluid passes through. The spatial discretization was realized via finite elements (FEM), resulting in an unstructured two-dimensional triangular mesh, as following: for the interior of the fluid domain, 1373 nodal points were used resulting in 2584 elements, whereas 160 elements were used for its boundaries. On the other hand, for the structure domain, 98 nodal points resulting in 144 elements were used for the interior and 32 elements for the boundaries. The time discretization was done via finite differences, using a step size of $10^{-4}$ s. Moreover, boundary conditions for the fluid solver are inflow for the left wall, with an initial velocity of 100, no-slip for top and bottom walls, and finally, outflow for the right wall. The configuration of the input physical parameters was of interest for the current work, whose initial values are outlined below.
5 Test Scenarios and Results

- density of the fluid $\rho_f = 0.01 \text{[kg/m}^3\text{]}$
- dynamic viscosity of the fluid $\mu_f = 0.03 \text{[kg/(m} \cdot \text{s)}\text{]}$
- density of the structure $\rho_s = 0.01 \text{[kg/m}^3\text{]}$
- Young’s modulus $E = 5\text{e}4 \text{[kg/(m} \cdot \text{s}^2\text{)]}$
- Poisson’s ratio $\nu_s = 0.3$

These parameters were at the heart of all UQ simulations and were considered either individually or in combinations as exhibiting uncertainty. Moreover, in all UQ simulations with normally distributed inputs, the mean of the inputs was equal to the corresponding deterministic values.

All simulations related to this scenario were performed using two machines from the Scientific Computing chair at Technische Universität München, one equipped with a dual core Intel Core2 Duo E7400 CPU, running at 2.8 GHz, and the other with an eight-core Intel Core i7 870 CPU, running at 2.93 GHz.

5.1.1 Deterministic Simulation

Before any UQ simulations were carried out, the first step was to perform deterministic simulations. In order to tackle the challenge of finding a suitable number of simulation time steps so that a single simulation is realistic from a computing and memory point of view, several scenarios were tested. Moreover, besides tackling the complexity of the solver, an additional challenge consisted of choosing the number of time steps such that the flow has time to develop and the structure to oscillate. Several numbers of time steps were simulated, namely 100, 200 and 500. The number that best fitted all the above criteria was 200. The resulting deterministic data was compared with all tested stochastic settings. The first and the last time steps of the deterministic simulation are depicted in Figure 5.2, where the visualized quantity was the pressure field of the fluid.

![Deterministic simulation at time-step 1](image1)
![Deterministic simulation at time-step 200](image2)

Figure 5.2: Deterministic simulation of the vertical flap
5.1 One Dimensional UQ Simulation

While the deterministic setting represented the first important step of the entire set of simulations, the first milestone for the UQ simulations was the one dimensional stochastic case. This means that throughout one simulation, four out of the five input physical parameters were taken to be sure or deterministic variables, i.e. defined to be exactly determined by given conditions ([33]), with the fifth modelled to exhibit uncertainty. Given the small number of input parameters, all five one-dimensional cases were simulated, paving the way for multidimensional settings. Furthermore, as from all UQ simulations the one-dimensional ones are the least complex, multiple simulation configurations were accounted for. The pseudo-spectral approach with a normally distributed input represented the core UQ methodology. The output at three locations of the FSI domain was evaluated in the post-processing step. The chosen locations were the right corner of the structure, i.e. the watch point outlined by preCICE, depicted in Figure 5.3b and two fluid locations, having coordinates 115 and 1147, depicted in Figure 5.3a. At the right corner, the considered QoI were the displacement on the x axis, the force on the x axis, and the force on the y axis. Furthermore, the QoI for the fluid locations were the pressure, velocity on the x axis, and velocity on the y axis. For the one-dimensional setting, MCS and the pseudo-spectral approach with an uniform random input were employed also when the investigated output was the right corner of the structure. The post-processing of the UQ simulations consisted of the evaluation of the mean (expectation) and the variance of the QoI at all time steps together with the estimation of the pdf at one chosen time step. Additionally, the computation of the mean and variance of the pressure and the displacement of the structure over the entire FSI domain was performed for a single time step.

In the first tested configuration, uncertainty was modelled as a normal random variable (n.r.v.) \( U_n \) of the form:

\[
U_n \sim N(\mu, \sigma^2), \text{ with } \sigma = 0.1\mu. \tag{5.1}
\]

In order to employ the pseudo-spectral approach, two parameters have to be specified, namely the quadrature degree, denoted by \( q \) and the number of coefficients, denoted by \( n \). Considering that the implementation provides functionality for \( q \) up to 10 (c.f. Section 4.2) and taking into account the rule of thumb suggested in [49], which proposes that \( q \leq 2n \), the employed combination in the current work was \( q = 8 \) and \( n = 5 \). Additionally, this strategy was also chosen because the higher the quadrature degree and number of coefficients, the lower the approximation error. The value of \( q \) implies that for each simulation
8 evaluations of the numerical solver were required. On the left side of Figure 5.4 the mean value for all QoI of the right corner location are depicted, for all five simulated scenarios. Additionally, the deterministic values are outlined. This plot shows that the mean values for all five stochastic settings are very similar to each other and with the deterministic result. An exception could be observed in the second half of the simulations for the x axis force, where the mean value corresponding to the Young’s modulus and density of the fluid are slightly different. From these observations, the following conclusions can be drawn. First, the similarity with the deterministic data points out that on average, the outcomes are very little affected by the input uncertainty. Second, besides the mean values of the x axis force corresponding to the two aforementioned parameters, the uncertainty in the other three inputs had a similar effect.

For a better understanding of the results, the variance plot depicted in the right part of Figure 5.4 was analysed next. Unlike the mean values, the variances corresponding to all input parameters were quantitatively different from each other. For all QoI the output corresponding to the Young’s modulus and density of the fluid had the highest variance. Moreover, in the case of the x axis displacement, the variance associated to the other three parameters is negligible, whereas for the forces, the variance resulted from uncertainty in Poisson’s ratio and density of the structure distinguished as well, but with a small magnitude. Note that the simulations in which the dynamic viscosity of the fluid was uncertain resulted in the lowest variance, which means that the uncertainty in this input parameter affected the outcomes the least. These results suggest that in multi-dimensional UQ scenarios, settings comprising Young’s modulus and the density of the fluid are expected to have the highest variance. Moreover, due to the negligible effect of the uncertainty...
5.1 Vertical Flap

in the dynamic viscosity of the fluid, this parameter was not considered in two-dimensional scenarios. The shapes of the variances are qualitatively similar to the ones of the means. However, the variance for the x displacement reaches its peak close to the end of the simulation, implying that the resulted uncertainty increases with time. Furthermore, the magnitude of the variance corresponding to the x axis force is larger than the one of the force on the y axis, meaning that the resulting uncertainty in the former is higher.

After the statistics evaluation, the next step consisted of the pdf estimation of the QoI using KDE. The estimation was performed for the UQ scenarios whose inputs were the density of the fluid and Young’s modulus, at time steps 20 and 190. In the KDE algorithm, 50000 random samples were employed. The results are outlined in Figures 5.5 and 5.6. Observe that at both time steps, the pdf corresponding to the scenarios having as input

![Figure 5.5: Pdf estimation at time step 20 when the inputs are the density of the fluid (left) and Young’s modulus (right)](image)

the density of the fluid is symmetric and Gaussian-like. This means that at these two time steps, the underlying value of the expectation provides a good estimate for the three QoI. Moreover, the numerical values of the mean and variance are sufficient to understand the impact of input uncertainty at these two time steps. On the other hand, the pdf estimate for the second input parameter is non-symmetric and negatively-skewed at time step 20, whereas at time step 190, the pdf is positively skewed. Note that for all three QoI, the pdf estimates have the same skewness. Thus, the pdf estimates offer additional information about the impact of uncertainty at the two time steps. Moreover, the non-symmetry of the pdfs implies that the mean value is not sufficient to provide an estimate of the QoI.

The results obtained via the pseudo-spectral approach were compared with the ones elicited by MCS with 100 samples. Their difference was quantified by calculating the mean
squared error\(^1\) (m.s.e.) of the means and variances for the case of uncertainty in density of the fluid. The error values are outlined in Table 5.1. Although only 100 samples were used for MCS, the error values are small. On one hand, they ascertain the results obtained via the pseudo-spectral approach with \(q = 8\) and \(n = 5\). On the other hand, they suggest that for this scenario MCS would not require a very large number of samples to achieve convergence. Similar results were obtained when testing for the other four parameters.

To get a better understanding of how the modelling of input uncertainty affects the outcomes, the pseudo-spectral approach was also employed together with inputs modelled as *uniform random variables* (u.r.v.) of the form \(U_u \sim U(a, b)\), with \(a = 0.9\mu\) and \(b = 1.1\mu\), where \(\mu\) is the mean used in the previous setting. This scenario was simulated when the stochastic input parameter was the density of the fluid. The comparison of the statistics is outlined in Figure 5.7. The expectation plot reveals strong similarities between the two strategies, hence, similar expected results, whereas the variances are quantitatively different, with higher variance corresponding to the normally distributed input. However, from a qualitative point of view the variances are similar, with the difference in magnitude being just a matter of scaling. The scaling factor can be explained by the fact that the support

---

\(^1\)for \(n\) measurements \(\hat{y}_i\) of \(y_i\), the mean squared error is defined as m.s.e. = \(\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2\)
for the input modelled as a n.r.v. is $\mathbb{R}$, with $\mu = 0.01$ and $\sigma = 0.001$, while the uniformly distributed case is supported by $[0.009, 0.011]$, with $\mu = 0.01$ and $\sigma = 0.00057^2$, therefore having a smaller variability. It can be concluded that the inputs modelled as u.r.v. do not provide new insight in the uncertainty analysis. Thus, for everything that follows, only normally distributed inputs were considered.

5.1 Vertical Flap

In the next step, the quantification of uncertainty was performed for the two coordinates of the fluid domain. In Figure 5.8, the expectations and variances are depicted for the QoI corresponding to the coordinate 115, whereas the statistics for coordinate 1146 are depicted in Figure 5.9. As in the previous case, the mean values are compared to the deterministic results. In both plots, the expectations are very similar to each other, as well as with the deterministic results. Even more, as in the case of the right corner of the structure, the expected values corresponding to the Young’s modulus and density of the fluid slightly differ from the rest for one QoI, i.e. the pressure. The similarity to the deterministic results ascertains the low impact of uncertainty in one dimensional scenarios. The similarity of the expected values for all stochastic simulations points out again that the effect of input uncertainty at the QoI was similar, regardless of the uncertain input.

In the variances plot it is observed again that the values differ for all five input parameters. Moreover, similarly to the structure scenario, some of the input parameters elicited higher variances. In the case of the first coordinate, these parameters were the Young’s modulus and the density of the fluid in the case when the QoI was the pressure, while in the case of velocities, additional to the two aforementioned parameters, the dynamic viscosity of the fluid exhibited a high variance. On the other hand, for the second coordinate,

$$2^2 \text{if } U_u \sim \mathcal{U}(a, b), \mu = \frac{a+b}{2}, \sigma = \sqrt{\frac{(b-a)^2}{12}} (47)$$
5 Test Scenarios and Results

Figure 5.8: Mean and variance of the QoI from coordinate 115 for all 1D UQ scenarios

Figure 5.9: Mean and variance of the QoI from coordinate 1147 for all 1D UQ scenarios
5.1 Vertical Flap

high variance has obtained when the uncertain inputs were the Young’s modulus and density of the fluid. From the five input parameters, uncertainty in the density of the solid and Poisson’s ratio seemed to output the smallest variance, implying that uncertainty in these two parameters has the least effect on the outcomes. For both fluid coordinates, the variance of the pressure has very high values and exhibits an oscillatory behaviour. Regarding the variance of the velocities, for the first coordinate the variance of the x axis velocity has an oscillatory behaviour, whereas the other grows at a fast rate when the inputs are the fluid density and Young’s modulus, and decreases when the input is the dynamic viscosity of the fluid. For the second coordinate, the variances of the velocities behave similarly, reaching their peak around time step 140. Note that their shape is correlated to the one of the displacement of the structure. i.e. when the displacement is higher, the uncertainty resulted at the QoI of this coordinate is higher as well.

The pdf estimation was done for the QoI of the fluid’s coordinate 115. Furthermore, the chosen time steps were identical to the ones employed in the case of the watch point, namely 20 and 190. The estimation was done for the scenarios where the input parameters were the dynamic viscosity of the fluid and Young’s modulus. The results are depicted in Figure 5.10 for time step 20, and in Figure 5.11 for time step 190. They are qualitatively similar to the ones from the case of the watch point, in the sense that the pdf estimate when the input is the dynamic viscosity of the fluid is symmetric, whereas the estimate for the case of the other parameter is skewed. Thus, the same conclusions are also valid here. However, note that the pdf estimate for the x axis velocity has opposite skewness compared to the other two. It follows that in both fluid and structure domains, at the chosen time steps, the influence of the uncertainty in Young’s modulus has a more significant impact than the uncertainty in the parameters of the fluid domain.

Figure 5.10: Pdf estimation at time step 20 when the inputs are the dynamic viscosity of the fluid (left) and Young’s modulus (right)
5 Test Scenarios and Results

Figure 5.11: Pdf estimation at time step 190 when the inputs are the dynamic viscosity of the fluid (left) and Young’s modulus (right)

Additionally to the statistics computation and pdfs estimation, statistics of QoI from the entire FSI domain were computed. In Figure 5.12, the means and variances of the pressure and the displacement of the structure are depicted. Their evaluation was performed at time step 190, when the uncertain input was Young’s modulus.

In the second tested configuration, uncertainty was again modelled as a normal random variable $U_n$, but with a higher variance, i.e. $U_n \sim N(\mu, \sigma^2)$, with $\sigma = 0.16\mu$. The statistics for the QoI of the right corner are outlined in Figure 5.13. It can be observed that the expectations are similar to each other and to the deterministic results, except for the values of the x axis force from the second part of the simulation, corresponding to scenarios having
as inputs the fluid density and Young’s modulus. Furthermore, the variances are qualitatively similar to the ones from the previous setting, but with a higher magnitude. These differences are accounted for by the increase in the input variance. From here, two important observations can be made. First, while the input uncertainty increases, so does the difference in the expectation values for the two aforementioned input parameters. Second, the increase in the input variance provides no additional information regarding the effect of uncertainty, compared to the first tested scenario. Similar results were obtained when the results for the QoI of the two fluid coordinates were compared. Hence, for everything that follows, uncertainty was modelled as in Equation 5.1.

![Figure 5.13: Mean and variance of the QoI corresponding to the second configuration](image)

5.1.3 Two Dimensional UQ Simulations

All two dimensional UQ simulations were performed via the pseudo-spectral approach with normally distributed random inputs. In the first step, two strategies were compared, i.e. a full grid approach using 64 points and a sparse grid based approach, using an interior grid of level 4 with 49 points together with a level 0 boundary grid with 4 points, hence with a total of 53 points. The purpose of their comparison was to understand the accuracy of the sparse grid based approach, given that the computational cost required by the two methods is similar.

Both strategies employed 15 expansion coefficients. Besides the statistics evaluation at each time step, which included the comparison of the expectations with the deterministic results, the uncertainty analysis in the two dimensional case comprised also a Sobol’s indices based sensitivity analysis at all time steps (c.f. Section 3.6).
The comparison of the two methodologies was performed for the QoI of the right corner of the structure, where the uncertain inputs were the density of the fluid and Young’s modulus. Even more, in the case of the sparse grids strategy, the interpolation of the solution on the interior grid was carried out using two types of basis functions: piecewise linear and piecewise polynomial of maximum degree two. The m.s.e. are outlined in Table 5.2, where s.g. denotes sparse grid. Although the errors are small, the highest value for the variance of the x axis force, obtained when the basis functions were the polynomials, is only in $O(10^{-1})$. Given that the two approaches have a similar computational cost, and that an increase in either the level of the interior or the boundary grid in the second approach would lead to a total number of points exceeding the one of the full grid, the tensor product rule was employed for the two dimensional UQ settings. This is further motivated by the goal of obtaining accurate UQ results.

The first post-processing step consisted of the analysis of the QoI corresponding to the right corner of the structure. Based on the results outlined by the one-dimensional simulations, the tested scenarios comprised the following uncertain inputs:

- $wp_1$: $\rho_f, E$
- $wp_2$: $\rho_f, \rho_s$
- $wp_3$: $\rho_f, \nu_s$
- $wp_4$: $E, \rho_s$
- $wp_5$: $E, \nu_s$
- $wp_6$: $\nu_s, \rho_s$

Due to the small variance exhibited in the one-dimensional scenario, dynamic viscosity of the fluid was not considered in the two dimensional case. The first five combinations are motivated by the high variance exhibited then the inputs were the density of the fluid and Young modulus in the one dimensional setting, while the last case involves two parameters whose propagation resulted in low individual variances. For simplicity, in what follows, the above input combinations are referred to by the corresponding id.

The statistics plot is outlined in Figure 5.14, where the expectations are on the left side and the variances on the right. The expectation plot show that the average behaviour of the stochastic results is similar to the deterministic case. However, combinations $wp_1$, $wp_3$, and $wp_6$ resulted in slightly different expectation values for the x axis force. Note that all three scenarios involve parameters who outlined a difference in the expectation of the x

<table>
<thead>
<tr>
<th>s.g. basis</th>
<th>statistic</th>
<th>x axis disp. m.s.e.</th>
<th>x axis force m.s.e.</th>
<th>y axis force m.s.e.</th>
</tr>
</thead>
<tbody>
<tr>
<td>linear basis</td>
<td>mean</td>
<td>1.185e-8</td>
<td>4.683e-3</td>
<td>2.171e-3</td>
</tr>
<tr>
<td></td>
<td>variance</td>
<td>1.560e-10</td>
<td>2.931e-1</td>
<td>6.222e-2</td>
</tr>
<tr>
<td>poly basis</td>
<td>mean</td>
<td>2.038e-6</td>
<td>8.229e-3</td>
<td>6.255e-3</td>
</tr>
<tr>
<td></td>
<td>variance</td>
<td>3.695e-10</td>
<td>9.905e-1</td>
<td>1.014e-1</td>
</tr>
</tbody>
</table>

Table 5.2: M.s.e of the sparse grid strategies compared to the full grid approach
axis force in the one dimensional case as well. In the variances plot, combination \( wp_1 \) resulted in the highest variance, while combination \( wp_6 \) in the lowest. Hence, based only on these two plots, it follows that little new information is gained from the two dimensional UQ simulations.

For a more detailed understanding of the resulted variance, in the next step the computation of the Sobol’s indices was performed. In Figures 5.15 and 5.16, Sobol’s indices are outlined for combinations \( wp_1 \) - the one with the highest overall variance, and \( wp_6 \), having the lowest overall variance. Note that \( S_1 \) denotes the index associated to the first parameter, while \( S_2 \) denotes the index associated to the second parameter. In addition, \( S_{12} \) denotes the contribution due to their interaction. It can be seen that in the case of x axis displacement, both \( S_{12} \) are close to zero, implying that the two input parameters behave independently. Hence the two dimensional simulations bring no new information. On the other hand, for the two axes forces the contribution of the inputs interaction to the total variance is non zero, meaning that they do not behave independently in the case of these QoI. To numerically quantify the overall contribution of the interaction index for each of the six scenarios, \( S_{12\,total} = \sum_{t=1}^{200} S_{12}(t) \) was computed. Note that as \( 0 \leq S_{12} \leq 1 \), it follows that \( 0 \leq S_{12\,total} \leq 200 \). The results are outlined in Table 5.3.

Table 5.3: Total interaction index for the 2D UQ simulations of the watch point
5 Test Scenarios and Results

Figure 5.15: Sobol indices for $\rho_f, E$

Figure 5.16: Sobol indices for $\rho_s, \nu_s$
Observe that all values corresponding to the x axis displacement are smaller than 0.5, meaning that in all considered two dimensional scenarios, the input parameters behave independently. On the other hand, for the two forces the highest values of the total interaction index were due to the combination \( wp_1 \), while the lowest was due to combination \( wp_2 \). As a conclusion, given the above results and the computed statistics, it follows that in case of the watch point, besides the combination involving the parameters that outlined the highest variance in the one dimensional setting, the two dimensional UQ scenarios offered little new information.

In the next step, the evaluation of the statistics for the two coordinates in the fluid domain was performed. The simulated scenarios included only combinations involving one of the parameters with high variance from the one dimensional case. For simplicity, they are enumerated below, having an assigned id used for their further identification:

- \( f_1: \mu_f, \rho_f \)
- \( f_2: \mu_f, \rho_s \)
- \( f_3: \rho_f, \nu_s \)
- \( f_4: \rho_f, E \)
- \( f_5: \mu_f, \nu_s \)

The results are depicted in Figures 5.17 and 5.18. The expectation plots corresponding to both fluid coordinates show a significant difference compared to the deterministic data for all three QoI. However, the stochastic results are similar to each other. Moreover, as
it is seen in Figure 5.18, the difference is more significant for the second coordinate. This implies that compared to all other tested scenarios, the impact of uncertainty is more substantial in this case. The difference of the expectations and deterministic data points out that in these scenarios the mean value does not provide a good estimator of the expected value of the outcome.

In the variances plot it can be seen that two combinations resulted in the highest variance, namely \( f_1, f_4 \). Moreover, it can be observed that the smallest variance was obtained for input combinations involving the density of the structure. Note that for the first coordinate, all variances present with an oscillatory behaviours. For the second coordinate, this is true only for the variance of the pressure. On the other hand, the variances of the velocities increases very rapidly once a threshold is reached.

![Figure 5.18: Mean and variance for the 2D simulations corresponding to coordinate 1147](image)

In the next step, the sensitivity analysis was performed. In this case, it was carried out for coordinate 115, when the inputs were \( f_1 \) and \( f_4 \). The results are depicted in Figures 5.19 and 5.20. Observe that in the two plots, the contribution of the interaction of the input parameters to the total variance was very insignificant, suggesting that the inputs behave independently. Moreover, in the case of the pressure, the contribution corresponding to one of the parameters was dominant. On the other hand, for the velocities, both inputs had significant contributions. To grasp an understanding of the total value of \( S_{12}, S_{12\text{ total}} \) was computed using the same approach as for the simulations regarding the right corner. The results are outlined in Table 5.4. Note that similarly to the case of the watch point, the values for \( S_{12\text{ total}} \) corresponding to one QoI are very small. In this scenario, this corresponds to the x axis velocity, pointing out that all two dimensional combinations behave almost independently. Moreover, for the pressure and y axis velocity, besides the combination \( f_4 \), all others presented with small values of \( S_{12\text{ total}} \).
5.1 Vertical Flap

Figure 5.19: Sobol indices for $\rho_f, \mu_f$

Figure 5.20: Sobol indices for $\rho_s, E$
5 Test Scenarios and Results

For both the watch point and coordinates of the fluid, the interaction of the density of the fluid and Young’s modulus had the most significant contribution to the resulted variance. Tacking into consideration all results corresponding to two dimensional UQ scenarios, it results that all tested scenarios could be reduced to the one involving the density of the fluid and Young’s modulus. For all other inputs, one dimensional UQ simulations would suffice.

Finally, the computation of the the statistics for the entire FSI domain was performed. In Figure 5.21, the mean and variance for the pressure and displacement of the structure are depicted at $t = 190$. Note that the statistics for the one dimensional scenario having Young’s modulus as uncertain input were depicted at time step 190, too. Interestingly, although the plots of the mean appears similar, the variance corresponding to the domain proceeding the structure is higher in the one dimensional scenario.

![Figure 5.21: Statistics of the entire FSI domain when the input combination comprises the density of the fluid and Young’s modulus](image)

5.1.4 Five Dimensional UQ Simulation

The five dimensional UQ simulation for the Vertical Flap scenario consisted of modelling all five input physical parameters as uncertain. The employed methodology was based on the pseudo-spectral approach and sparse grids interpolation. However, to ascertain the results obtained via the aforementioned approach, MCS was also run once, with 500 samples. A number of five interaction terms was used for each dimension, resulting in a total of 126 expansion coefficients. Given the 200 time steps for one simulation, 25200 coefficients were computed in total.

Unlike the previous two settings where the post-processing step included the analysis
of both the watch point and fluid coordinates, in the five dimensional setting the focus was only on the watch point. Besides the high complexity of the underlying setting, this choice was also motivated by the fact that additionally to the quantification of uncertainty, another end goal was to find a suitable combination of levels for the interior and boundary grids so that the total number of interpolation points needed for obtaining accurate results was as small as possible. To this extent, level 3 and 4 interior grids were combined with level 0 and 1 boundary grids for realizing the interpolation of the numerical solution. For the interior grid, both piecewise linear and polynomial of degree 2 basis functions were employed.

At the end of the simulations, the computed quantities were the mean and the variance of the QoI associated to the watch point together with six local Sobol’s indices, five corresponding to each input parameter and the sixth to their total interaction. The purpose of the latter was to see whether the stochastic dimensionality could be reduced and, furthermore, which parameters provided the most significant contribution to the total variance. Moreover, the expectations are compared to the deterministic counterparts.

The first step consisted of finding a suitable combination of levels for the two grids used for interpolation. To this extent, several combinations were tested. The end results were compared with the combination comprising the most points, for which piecewise polynomial basis functions of degree two were used for the interior grid. The comparison was performed by computing the m.s.e. The considered combinations are listed below, together with the used basis functions for the interior grid:

- \( \text{comb}_1 \): level 3 interior (71 points), level 1 boundary (32 points), poly basis deg 2
- \( \text{comb}_2 \): level 4 interior (381 points), level 1 boundary (112 points), linear basis
- \( \text{comb}_3 \): level 4 interior (381 points), level 0 boundary (32 points), poly basis deg 2
- \( \text{comb}_4 \): level 4 interior (381 points), level 1 boundary (112 points), poly basis deg 2

The results are presented in Table 5.5. It can be seen that the highest errors corresponding to the statistics computed for the x axis force. Moreover, note that for the first two strategies, the error for the variance was in \( O(1) \). On the other hand, when the third strategy was employed, the errors dropped by an order of two, with the highest being in \( O(10^{-2}) \).

Given the difference of 80 interpolation points, i.e. 80 calls of the numerical solver between

<table>
<thead>
<tr>
<th>strategy</th>
<th>total points</th>
<th>statistic</th>
<th>x axis disp. m.s.e.</th>
<th>x axis F m.s.e.</th>
<th>y axis F m.s.e.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{comb}_1 )</td>
<td>183</td>
<td>mean</td>
<td>1.602e-7</td>
<td>5.654e-2</td>
<td>2.795e-2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>variance</td>
<td>4.665e-10</td>
<td>2.573e0</td>
<td>6.448e1</td>
</tr>
<tr>
<td>( \text{comb}_2 )</td>
<td>463</td>
<td>mean</td>
<td>2.669e-9</td>
<td>2.891e-2</td>
<td>1.787e-2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>variance</td>
<td>3.012e-10</td>
<td>5.960e0</td>
<td>7.967e-1</td>
</tr>
<tr>
<td>( \text{comb}_3 )</td>
<td>383</td>
<td>mean</td>
<td>2.736e-7</td>
<td>1.526e-2</td>
<td>6.834e-3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>variance</td>
<td>8.139e-13</td>
<td>1.872e-2</td>
<td>3.037e-3</td>
</tr>
</tbody>
</table>

Table 5.5: M.s.e. of the sparse grid strategies compared to the full grid approach

strategies \( \text{comb}_3 \) and \( \text{comb}_4 \), the former was adopted in the current work for the testing of the five dimensional UQ setting.
In order to validate the chosen strategy for the pseudo-spectral approach, the results were compared to the out of MCS, employing 500 samples. The m.s.e. are outlined in Table 5.6. Observe that all errors are at least in $O(10^{-1})$. Moreover, considering that only 500 samples were employed for MCS, it follows that difference between the results obtained via MCS and the pseudo-spectral approach is small enough to state that the combination of sparse grids interpolation and numerical quadrature was appropriate to tackle the high dimensional UQ problems.

The resulted statistics are depicted in Figure 5.22, where the expectation is on the left side of the figure, whereas the variance is on the right. Moreover, the expectations are compared with the deterministic data. Observe that the mean of the x axis displacement and the deterministic data are very similar to each other. Furthermore, except for the second half of the simulation, the expectation of the y axis force is similar to the deterministic result. On the other hand, for the x axis force, the expectation of the stochastic results is significantly different from the deterministic counterpart, starting with about time step 60. This implies that the impact of uncertainty in the inputs was higher for the x axis force. The variances

<table>
<thead>
<tr>
<th>statistic</th>
<th>x axis displacement m.s.e.</th>
<th>x axis force m.s.e.</th>
<th>y axis force m.s.e.</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean</td>
<td>2.719e-6</td>
<td>2.722e-2</td>
<td>1.785e-2</td>
</tr>
<tr>
<td>variance</td>
<td>4.495e-11</td>
<td>4.803e-1</td>
<td>1.753e-1</td>
</tr>
</tbody>
</table>

Table 5.6: M.s.e. between MCS and the pseudo-spectral approach

Figure 5.22: Mean and variance for the 5D UQ simulations obtained with the pseudo-spectral approach
5.1 Vertical Flap

have similar shapes to the results obtained for one and two dimensional UQ settings. Note that their magnitude is similar to the one of the variances from the two dimensional setting.

In Figure 5.23 the six computed local Sobol’s indices are depicted. Moreover, for a deeper understanding of the results, the total value of each index was computed for the entire simulation time span, as $S_{i \text{tot}} = \sum_{t=1}^{200} S_i(t), i = 1, \ldots, 6$, where $0 \leq S_{i \text{tot}} \leq 200, i = 1, \ldots, 6$. The results are outlined in Table 5.7.

![Figure 5.23: Local Sobol’s indices for the 5D UQ scenario](image)

<table>
<thead>
<tr>
<th>x axis disp</th>
<th>$S_1 \text{tot} (\nu_f)$</th>
<th>$S_2 \text{tot} (\rho_f)$</th>
<th>$S_3 \text{tot} (\rho_s)$</th>
<th>$S_4 \text{tot} (E)$</th>
<th>$S_5 \text{tot} (\nu_s)$</th>
<th>$S_6 \text{tot} (\text{inter.})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>x axis force</td>
<td>7.825</td>
<td>76.497</td>
<td>26.293</td>
<td>87.027</td>
<td>1.640</td>
<td>0.721</td>
</tr>
<tr>
<td>y axis force</td>
<td>6.209</td>
<td>61.578</td>
<td>8.160</td>
<td>81.447</td>
<td>6.359</td>
<td>36.249</td>
</tr>
</tbody>
</table>

Table 5.7: Contribution of the input parameters and their interaction to the overall variance on the portions where the variance of the QoI has significant values

Analysing both the plot and the values from the table, it can be seen that the parameters whose variance contributes the most to the total variance are the density of the fluid and Young’s modulus. For all three QoI, their combined contribution is around 75% of the total value. The logical conclusion would be that out of the five parameters, only these two were relevant for the current UQ scenario. This would imply that the UQ problem could be reduced from dimension five to two. However, given that the interaction indices for the forces have significant values, the validity of the aforementioned conclusion cannot be fully ascertained. To this extent, in the next step, the percentage of the total interaction...
index due to the combination of the two parameters was computed. The results were 73.63\% for the x axis force and 65.01\% for the y axis force. Thus, it follows that indeed the variances due to the density of the fluid, Young’s modulus, and their interaction contribute dominantly to the total resulted variance. Hence, the five dimensional UQ problem could be reduced to a two dimensional version.

5.2 FSI Benchmark Scenario

The second tested FSI scenario was a benchmark scenario, proposed in [50], with the name FSI3. If the first application served as a model problem, the underlying scenario plays the role of a surrogate for typical real-world FSI applications. To this extent, uncertainty was quantified only in a five-dimensional setting, with the quantities of interest being measured at one specific location.

Broadly speaking, this scenario comprises the flow of an incompressible, Newtonian fluid, interacting with an elastic solid. The domain is based on the two dimensional version of the Computational Fluid Dynamics (CFD) benchmark proposed in [51], which is depicted in Figure 5.24. Furthermore, the structure part is outlined in Figure 5.25. The comparison with the older flow benchmark can be easily achieved by omitting the elastic bar behind the cylinder. In this way, the flow around a cylinder configuration is exactly recovered ([50]).

The prescription of the geometry parameters is outlined below.

- domain dimensions are \( L = 2.4 \) and \( H = 0.41 \)
- measured from the left corner of the channel, the circle’s center is \( C=(0.2, 0.2) \) and radius \( r = 0.05 \)
- elastic structure’s length \( l=0.35 \) and height \( h=0.02 \)
- elastic structure’s right bottom corner positioned at \( (0.6, 0.19) \), with the left end being fully attached to the fixed cylinder
- control points are \( A(t) \), with \( A(0) = (0.6, 0.2) \) and \( B = (0.15, 0.2) \)
It is important to remark that in this work, the point at coordinates $(0.6, 0.2)$ was used as the location where the UQ specific measurements were performed. Moreover, as it can be observed from above description, the setting is non-symmetric. This was intentional, with the purpose of preventing the dependence of the onset of any potential oscillation on the precision of the computation ([50]).

The boundary conditions are prescribed as follows: at the left channel inflow, a parabolic velocity profile is prescribed, of the form:

$$v_f(0, t) = 1.5\bar{U} \frac{4.0}{0.1681^2} y(0.41 - y),$$

(5.2)

where $\bar{U}$ denotes the mean flow velocity. With this expression, the maximum of the inflow is limited to $1.5\bar{U}$. The outflow condition effectively prescribes a reference value of the pressure and for this benchmark scenario, it is user defined (e.g. stress free or do nothing conditions). Moreover, on the top, bottom, circle and fluid-structure interface, no-slip conditions are prescribed. The suggested initial conditions were chosen so as to use a smooth increase of the velocity profile in time:

$$v_f(t, 0, y) = \begin{cases} v_f(0, y)\frac{1-\cos(\pi t/2)}{2}, & \text{if } t < 2.0, \\ v_f(0, y), & \text{else} \end{cases}$$

(5.3)

where $v_f(0, y)$ is the velocity profile given in Equation 5.2.

In Alya, the spatial discretization of the benchmark was realized as follows: for the fluid domain, 35524 nodal points were used resulting in 69460 two dimensional triangle elements, whereas at the boundary, 1588 elements were employed. For the domain of the structure, 8295 nodal points were used, resulting in 15850 triangle elements for the interior and 738 elements for the boundary. Furthermore, the step size used for the finite differences discretization in time was $10^{-3}$ s.

In the set-up files of Alya, five physical input parameters are specified. For a deterministic run, their specific values found in [50], are:

- Density of the fluid $\rho_f = 1.000$ [$kg/m^3$]
- Dynamic viscosity of the fluid $\mu_f = 1.0$ [$kg/(m \cdot s)$]
- Density of the structure $\rho_s = 1.000$ [$kg/m^3$]
- Young’s modulus $E = 5.6e5$ [$kg/(m \cdot s^2)$]
- Poisson’s ratio $\nu_s = 0.4$

Moreover, $\bar{U} = 2[m/s]$ and $Re = 200$.

All deterministic and UQ simulations were run on the Munich Center of Advanced Computing’ MAC Cluster, using nodes from the snb and bdz partitions$^3$. The first partition features 28 dual socket Intel Sandy Bridge-EP Xeon E5-2670 nodes, equipped with 128 GB of RAM, while the bdz partition features 19 quad socket AMD Bulldozer Opteron 6274 nodes, equipped with 256 GB of RAM. In both partitions, the communication is realized

$^3$see http://www.mac.tum.de/wiki/index.php/MAC_Cluster$#$Hardware for more details
via Quadruple Data Rate (QDR) infiniband. Additionally, the Intel compiler toolkit (icc and Intel MPI) was employed. Due to the high complexity of a single simulation, only the method based on discrete projections and sparse grids was used for this scenario. Moreover, a single simulation was run using the parallel version of the multi-physics solver, Alya, together with a socket-based coupling realized by preCICE. Global parallelism, i.e. parallelism at the level of to the entire set of simulations, was achieved using the batch system on the MAC Cluster, where multiple simulations were launched at the same time, with the remark that a maximum number of 10 jobs is allowed for one user within a single day. In order to keep a balance between the large number of complex simulations needed to be executed, an individual simulation was run on a single node, where, having available a total of 16 tasks, 12 were used for the fluid solver and 4 for the solver of the structure. Note that for memory considerations, the binary files yielded by Alya were saved only every 100 time steps.

5.2.1 Deterministic Simulations

Before the five dimensional UQ simulation was performed, the first step was to carry out one deterministic simulation. Unlike the previous scenario, where several numbers of time steps were simulated, in this case the simulation was let to run for 24 hours to see how many time steps could be performed within this time limit. Employing one Sandy Bridge node from the MAC Cluster, 6639 time steps were performed, hence 6.639 seconds of simulation. Note that this number does not necessarily correspond to the number of iterations of the entire UQ simulation. Depending on whether the Sandy Bridge or Bulldozer nodes are employed and the necessary number of iterations to achieve convergence, the number of performed time steps differs from simulation to simulation. The FSI domain of the benchmark is depicted at time step 6639 in Figure 5.26, where the physical quantity from the fluid domain is the velocity field.

![Figure 5.26: FSI benchmark at time-step 6639](image)

5.2.2 Five Dimensional UQ Simulation

The UQ simulation for the FSI benchmark was performed by considering all five input parameters as uncertain. Moreover, they were modelled as normal random variables $U_n$ of the form:

$$U_n \sim \mathcal{N}(\mu, \sigma^2), \text{ with } \sigma = 0.1\mu,$$

(5.4)
there the value of $\mu$ was the corresponding deterministic value.

The employed UQ methodology was based on the pseudo-spectral approach with sparse grids interpolation. Due to the high complexity of the benchmark, a challenging task was to find a suitable number of interpolation points. The employed strategy comprised a level 3 grid for the interior, consisting of 71 points and a level 0 grid for the boundary, comprising 32 points. Thus, a total of 103 interpolation points were used. The chosen basis functions for the interior grid was based on piecewise polynomials of degree 2. Even though this number seems small for a typical five dimensional problem, remember that a single simulation was ran for 24 hours. Moreover, the end of the entire UQ simulation resulted in more than 1 TB of output data. To keep the numerical error due to quadrature as small as possible, the employed degree for the one-dimensional Gauss-Legendre rule (c.f. Equation 3.51) was 20. Moreover, the number of interacting terms in one dimension was set to five. Thus, the size of the orthogonal basis employed in the spectral expansion was 126.

In the post-processing step, the analysed QoI were the displacement delta on the x axis, displacement delta on the y axis, and the x axis force corresponding to the tip of the structure, i.e. point A(0.6, 0.2) (c.f. Section 5.2). The computed quantities were the mean and the variance together with the Sobol’s indices used to perform the variance based sensitivity analysis. In the latter, besides the computation of the indices, their overall contribution was also calculated. Moreover, the value of the expectations was compared with the deterministic data. Additionally, for a complete description of the UQ simulation, the execution times for pre-processing, simulation, and post-processing were also measured.

As stated in Section 5.2.1, similar to the deterministic simulation, all 103 UQ simulations were run for 24 hours. Tacking into account the restriction of maximum 10 jobs per partition and the occupancy of the MAC cluster during the period of time when the simulations were performed, the entire set of simulations was finished in a time span of eight days. The resulted number of simulated time steps ranged from 4000 to 7000. This implied that the number of simulation time steps used in post-processing was the minimum value, i.e. 4000. Note that as the size of the five dimensional basis is 126, a total of 504000 coefficients were computed. Because the computed coefficients are saved in a text file at each time step, as the number of the saved coefficients increases, so does the size of the text file. In order to minimize the time required for opening the file and saving the coefficients, four chunks consisting of $1000 \cdot 126 = 126000$ coefficients each were computed and saved separately. At the end, they were merged in one file, consisting of all coefficients.

The execution time for the three steps of the UQ simulation is outlined in Table 5.8. Note that by post-processing, it is meant the computation of the coefficients of the spectral expansion. After the coefficients were obtained, the computation of statistics and sensitivity indices was immediate, requiring an insignificant computational cost. It follows that the

<table>
<thead>
<tr>
<th>UQ simulation step</th>
<th>execution time (h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>pre-processing</td>
<td>0.09</td>
</tr>
<tr>
<td>simulation</td>
<td>192</td>
</tr>
<tr>
<td>post-processing</td>
<td>1.4</td>
</tr>
</tbody>
</table>

Table 5.8: Execution times of the constituent parts of the 5D UQ simulation
cost of the simulation part was more than 128 times more expensive than the cost of the pre- and post-processing combined.

In Figure 5.27, the mean (on the left side) and the variance (on the right side) of the QoI are depicted. Moreover, the plot of the mean includes the deterministic results as well. It can be observed that for more than 3600 time steps, the expectations and the de-

![Figure 5.27: Mean and variance for the 5D UQ scenario](image)

terministic values are very similar to each other and have low values. However, through the end of the simulated time frame, both expectations increase, exhibiting an oscillatory behaviour. Considering that the corresponding deterministic values remain small during this time frame, the difference implies that the input uncertainty had a significant impact on the outcomes. On the other hand, the UQ and deterministic results for the x axis force are qualitatively similar; in the first quarter of the simulation, its values are smooth and afterwards, they begin to oscillate. However, quantitatively, in the time frame 2000-4000, their amplitude is different. For the deterministic data, the amplitude is almost constant, whereas the amplitude of the expectation of the x axis force decreases until about time step 3000, starting to increase afterwards, with a higher rate when the values of the displacements deltas expectation start to also oscillate.

The variances plot depicts results that are qualitatively correlated to their corresponding means. For both displacement deltas, the variances are close to zero when their mean and the deterministic data are similar, and starts to grow when the magnitude of the means increase. Moreover, for the x axis force, the variance is very small in the same region where its expectation is similar to the deterministic result and starts to grow when the mean value begins to oscillate.

Based on these results, it follows that in the case of the two displacement deltas, the effect of uncertainty is insignificant until about time step 3600. Moreover, this holds true also for the x axis forces, but starting earlier, at about time step 1100.
To get a better understanding of the resulted uncertainty, the sensitivity analysis was performed next, which consisted of the computation of the local Sobol’s indices corresponding to each input parameter together with the index associated to their interaction. The result is depicted in Figure 5.28. On the portion where the variance corresponding to the displacement deltas is insignificant, the contribution of the individual parameters and their interaction follows a similar pattern. In the first 500 time steps, the contributions of the density of the fluid and Young’s modulus seem to be the most significant. Although unrelated, this is similar to what was obtained for the previous scenario. After this time step, the contribution is shared between all input parameters and their interaction. Note that the contribution of the density of the fluid seems to be the least significant in this entire time frame. The paradigm shifts on the portions where the variances have significant values. In the case of the x axis displacement delta, this portion distinguishes from the rest by the fact that the major contributions to the overall variance are due mostly to Poisson’s ratio, Young’s modulus, and, seldom to the density of the fluid. In addition, on these portion, the contribution due to the parameters interaction decreases. A similar qualitative pattern holds for the indices corresponding to the y axis displacement. On the portion where the resulted variance is small, the major contribution is due to the Young’s modulus, dynamic viscosity of the fluid, and the parameters interaction. When the variance begins to grow, the influence of the parameters interaction decreases, whereas the major contribution is due to the density of the fluid and Young’s modulus. On the other hand, for the x axis force, in the first part of the simulation, the major contribution to the total variance is mainly due to the two parameters of the fluid domain. When the variance increases, the contribution due to the dynamic viscosity of the fluid and the interaction of parameters increases as well. Based on this plot alone, it can be inferred that all parameters have a significant contribution to the total variance.
5 Test Scenarios and Results

Given the observations drawn from the plot of the Sobol’s indices, it would be interesting to compute the contribution of each input parameter and of their interaction to the overall variance on the portions where the variance has significant values. This translates into the computation of:

$$\tilde{S}_i = \sum_{t=a}^{b} S_i(t), \quad i = 1, \ldots, 6,$$

where $a$ and $b$ are the extremities of the interval of interest. To this extent, the time intervals 3700-4000, 3550-4000 and 1400-4000 were chosen for the three QoI respectively, to perform the computations. The results are outlined in Table 5.9. In the above table it can be seen

<table>
<thead>
<tr>
<th></th>
<th>$\tilde{S}_1 (\nu_f)$</th>
<th>$\tilde{S}_2 (\rho_f)$</th>
<th>$\tilde{S}_3 (\rho_s)$</th>
<th>$\tilde{S}_4 (E)$</th>
<th>$\tilde{S}_5 (\nu_s)$</th>
<th>$\tilde{S}_6$ (inter.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>x axis displacement</td>
<td>51.629</td>
<td>22.100</td>
<td>6.062</td>
<td>57.549</td>
<td>129.112</td>
<td>31.550</td>
</tr>
<tr>
<td>x axis force</td>
<td>13.531</td>
<td>63.086</td>
<td>11.235</td>
<td>314.717</td>
<td>15.053</td>
<td>30.381</td>
</tr>
<tr>
<td>y axis force</td>
<td>820.902</td>
<td>274.448</td>
<td>183.507</td>
<td>629.728</td>
<td>48.824</td>
<td>641.593</td>
</tr>
</tbody>
</table>

Table 5.9: Contribution of the input parameters and their interaction to the overall variance on the portions where the variance of the QoI has significant values

that for the total variance of all three QoI, the parameters that give the major contribution are from the domain of the structure - for the displacements, and the dynamic viscosity of the fluid - for the x axis force. Moreover, the contribution due to the interaction of the parameters is small in the case of the displacements, whereas for the force is very significant. Different to the previous scenario, where the influence of two input parameters was significant in the majority of UQ simulations, in this scenario all parameters together with their interaction are significant for the total variance.
6 Conclusions

This work presented the problem of combining fluid-structure interaction with uncertainty quantification, via forward propagation of uncertainty. It represented a proof of concept that tried to answer questions such as whether uncertainty quantification is important for fluid-structure interactions, whether the applied methodology was the appropriate one, or whether the boundaries of the resulting problem can or cannot be overreached. The stochastic modelling was performed for five input physical parameters: density and dynamic viscosity of the fluid, density of the structure, Young’s modulus, and Poisson’s ratio.

Due to the multi-challenge posed by the resulting problem, high performance computing and advanced numerical methods were used. The multi-physics and multi-domain challenges were tackled by employing state of the art simulation codes, i.e. Alya and preCICE, with the latter realizing the coupling of the multi-physics components via the partitioned approach. Uncertainty quantification was performed by employing the pseudospectral approach, with the Monte Carlo algorithm being used only for the validation of the results. Moreover, high dimensional stochastic problems, which in essence were multi-dimensional quadrature problems, were approached via sparse grids interpolation combined with numerical integration. This idea was possible because of the assumption that the stochastic inputs are independent. However, given that uncertainty was modelled in the input physical parameters, the independence assumption made perfect sense. To keep a balance between the computational cost and accuracy, a combination of two grids was used for the interpolation problem: one for the interior and the other for the boundary of the domain. The sparse grids functionality was provided by the SG++ library. Furthermore, the multi-core challenge was tackled by performing all simulations using two levels of parallelism: a local one, at the level of a single simulation and a global level of parallelism, at the entire set of simulations.

In post-processing, a broad spectrum of functionality was available, including the computation of the statistics of quantities of interest, either from a specific location of the domain, or at the level of the entire domain. Moreover, it also comprised the estimation of the probability density function, and a Sobol’s indices based sensitivity analysis. The latter had the purpose to reveal the contribution of each input parameter together with their interaction to the totally resulted uncertainty.

The simulation part employed two fluid-structure interaction applications. The first one was a two dimensional wall-mounted flap and it was tested for stochastic dimensionality one, two, and five. The post-processing of the first two settings showed that two out of the five input parameters contributed the most to the resulted uncertainty. Moreover, the analysis of the five dimensional scenario confirmed that those two parameters together with their interaction contribute the most to the total variance, implying that the five dimensional problem could be reduced to a two dimensional one. The second tested scenario was a fluid-structure interaction benchmark proposed by S. Turek et al. in ([50]), for which the considered stochastic dimension was five. The post-processing revealed a significant
impact of uncertainty in the superior region of the simulated time span, and, moreover, that the contribution of all input parameters together with their interaction was pivotal to the overall resulted uncertainty.

The end results proved that uncertainty quantification is important when testing fluid structure interaction problems, as uncertainty in the inputs had a big impact in the outcomes. Moreover, sensitivity analysis turned out to be a very useful tool for finding not only how the inputs contribute to the overall resulted uncertainty, but also whether the dimensionality of the problem could be reduced. The curse of dimensionality in high dimensional stochastic problems was successfully tackled by the approach comprising the pseudo-spectral approach and sparse grids interpolation. The combination of interpolation and quadrature was very effective, even when the total number of interpolation points was small. Even more, due to the non-intrusiveness of the methodology, all simulations needed to be performed only once, keeping the required computation cost at the minimum. The separation of pre-processing, simulation, and post-processing provided an easier way to perform simulations and paved the way for employing two levels of parallelism, which is quintessential for tackling problems with a high level of complexity.

The heart of the pseudo-spectral approach was the computation of the spectral expansion coefficients whose computational cost was mainly influenced by the cost of the underlying numerical solution. Hence, should sufficient computational resources be available, the first idea of extension would be to employ an increased number of sparse grids interpolation points, either for the interior grid or for the boundary grid. Moreover, another possibility is to employ an alternative interpolation method, based on adaptive sparse grid. Furthermore, an approach that proved to an optimal choice for dealing with high dimensional applications is the combination technique ([28]), which, for uncertainty quantification, could be used for the evaluation the high dimensional integrals.

The non-intrusiveness of the employed methodology implies that it can be easily adapted for problems with stochastic dimensionality higher than five. However, note than when the dimensionality is high enough so that the resulted problem still suffers from the curse of dimensionality, the Monte Carlo algorithm is a better choice, as its convergence rate is dimension independent. If uncertainty is considered in the boundary conditions and/or other constituents of the input which are continuous quantities, the current methodology can still be applied. The difference would consist of modelling the input uncertainty not as a random variable, but as a random field, whose parametrization is typically done via the Karhunen-Loeve expansion, spectral series, or orthogonal series ([47]). An interesting question would be what happens when three dimensional multi-physics problems are considered (e.g. blood flow simulations). In that case, the stochastic methodology would not change, as it treats the underlying system as a black box. However, the computational cost would be much greater, so appropriate computational resources are required. Note that even if the computational demands are satisfied, the memory requirements still represent a challenge. A possible solution would be to employ an approach similar to one in the post-processing of the second scenario: the simulation and post-processing parts could be divided into smaller chunks, and immediately after a batch of simulations is done, post-processing is performed, and, at the end, all results are merged.

Nevertheless, especially in the context of complex problems, uncertainty quantification is a very active scientific branch, where a significant research potential exists. To this extent, in the future, the author plans to test and implement all the aforementioned ideas.
Bibliography


Bibliography


