Data-driven uncertainty quantification for high-dimensional engineering problems

C. Lataniotis

DISS. ETH NO. 26395

DATA-DRIVEN UNCERTAINTY QUANTIFICATION FOR HIGH-DIMENSIONAL ENGINEERING PROBLEMS

A dissertation submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH
(Dr. sc. ETH Zurich)

publicly presented and defended on 8 November 2019 by

CHRISTOS LATANIOTIS
MSc. ETH in Robotics, Systems and Control
Dipl. NTUA in Mechanical Engineering

born on 1 March 1985
citizen of Greece

accepted on the recommendation of

Prof. Dr. B. Sudret, supervisor
Dr. S. Marelli, co-supervisor
Prof. Dr. E. Chatzi, examiner
Prof. Dr. J.-M. Bourinet, examiner

2019

DOI: 10.3929/ethz-b-000377865

To Eleni

ABSTRACT

In the context of complex industrial systems and civil infrastructures, taking into account uncertainties during the design process has received much attention in the last decades. Although there is significant progress in modelling such systems, there are always discrepancies between ideal in-silico designed systems and real-world manufactured ones.

Starting from a realistic computational model that reproduces the behaviour of an engineering system, uncertainty quantification aims at modelling the various sources of uncertainty (including natural variability and lack of knowledge) affecting its input parameters as well as propagating these uncertainties to the response quantities of interest (*e.g.* performance indicators). Due to the high-fidelity and related computational costs of such models, the use of Monte Carlo methods for uncertainty quantification is often not a viable solution. To overcome this limitation, the use of surrogate models has become well established. A surrogate model is an analytical function that provides an accurate approximation of a computational model, based on a limited number of runs of the simulator at selected values of the input parameters and an appropriate learning algorithm.

In this thesis, the focus is the application of modern uncertainty quantification techniques in the presence of a large number, up to several thousands, of system parameters. As the dimensionality of the input space increases, the performance of surrogate modelling methods decreases, an issue that is known as *curse of dimensionality*. Furthermore, we approach the problem from a purely data-driven perspective, *i.e.* the entire analysis needs to be conducted based only a limited number of observations and little to no assumptions about the inner workings of the system. This scenario has high practical relevance, *e.g.* due to complex workflows involving various software packages to simulate a system or real-world applications for which only measurements of the input parameters and model responses are available. However such data-driven approaches introduce additional challenges related to the (unknown) stochastic properties of the input space. To quantify those, one typically resorts to well-known inference techniques

(discussed in Chapter 2), but such methodologies also suffer from the curse of dimensionality.

To enable data-driven uncertainty quantification in high-dimensional input spaces, we propose a combination of machine learning techniques for data compression and state-of-the-art surrogate modelling introduced by the uncertainty quantification community. The first fundamental ingredient, dimensionality reduction, is discussed in Chapter 3. Through a literature review on the rather broad topic of dimensionality reduction, we highlight the strengths and weaknesses of various techniques as well as their area of application.

The second fundamental ingredient, surrogate modelling, is discussed in Chapter 4. Beyond a general formulation, focus is given on two state-of-the-art techniques, namely Kriging and polynomial chaos expansions, that are used throughout this thesis.

A novel methodology for enabling surrogate modelling in high dimensional spaces is introduced in Chapter 5. The proposed algorithm couples the input compression and surrogate modelling steps in such a way that the resulting performance of the surrogate is optimal. Furthermore we demonstrate its consistently superior performance on several benchmark applications (in terms of the predictive accuracy of the surrogate), compared to traditional approaches that treat dimensionality reduction and surrogate modelling as two disjoint steps.

In Chapter 6, we propose a workflow for data-driven uncertainty quantification in high dimensional spaces, which is the ultimate goal of the thesis. The proposed workflow capitalises on the findings of the previous Chapters. Having access to a compressed space of manageable size and a surrogate, we show how one can eventually calculate statistical properties of the quantities of interest, such as their moments, quantiles and even their full probability distribution function. After applying this methodology on benchmark applications, we demonstrate that this workflow can lead to improved estimates of the uncertainty of the quantities of interest, especially in the extreme value regions.

Finally, in Chapter 7 we show how the methods presented in this thesis can be applied to a realistic engineering application related to the structural health monitoring of wind turbines. The goal is to estimate the fatigue

accumulation and peak loads, as well as their uncertainty, on various components of a wind turbine, given the inflow wind speed over 10 minute time intervals. We do so by processing a limited amount of observations that are generated by specialised software.

This manuscript introduces new techniques that enable uncertainty quantification in a wide class of problems for which it was initially not possible. This has strong practical implications considering the numerous relevant problems nowadays in *e.g.* structural health monitoring, earthquake engineering, weather forecasting, hydrogeology and control engineering, where the input space is high-dimensional (*e.g.* time series or image inputs). The new methodology and our findings are summarised in Chapter 8, along with suggestions for future research on this topic.

-

ZUSAMMENFASSUNG

Die Berücksichtigung von Ungewissheiten im Designprozess von komplexen industriellen Systemen und ziviler Infrastruktur hat in den letzten Jahrzehnten viel Beachtung gefunden. Trotz des beträchtlichen Fortschritts im Modellieren solcher Systeme gibt es immer Diskrepanzen zwischen den idealen, in-silico entwickelten Systemen und ihren Realisierungen.

Ausgehend von einem genauen Computermodell, das das Verhalten eines technischen Systems simuliert, zielt die sogenannte Ungewissheitsquantifizierung darauf ab, die verschiedenen Ungewissheitsquellen der Eingabeparameter, zu denen auch Unkenntnis und natürliche Variabilität der Parameter gehören, zu modellieren sowie die Fortpflanzung dieser Ungewissheiten in die Ergebnisse des Computermodells zu untersuchen (z.B. durch die Auswertung von Leistungsindikatoren). Wegen der hohen Genauigkeit und dem damit verbundenen Rechenaufwand ist die Verwendung von Monte Carlo-Methoden für Ungewissheitsquantifizierung oft nicht möglich. Eine bekannte Methode, diese Einschränkung zu umgehen, ist die Verwendung sogenannter Ersatzmodelle. Ein Ersatzmodell ist eine analytische Funktion, die ein Computermodell präzise annähert, indem eine begrenzte Anzahl von Durchläufen dieses Computermodells an ausgewählten Eingabewerten und ein geeigneter Lernalgorithmus verwendet werden.

Der Fokus dieser Arbeit liegt auf der Anwendung moderner Verfahren zur Ungewissheitsquantifizierung auf Computermodelle mit einer sehr hohen Anzahl (d.h., bis zu mehreren tausenden) von Eingabeparametern. Mit steigender Dimensionalität des Eingaberaums sinkt die Leistung der Ersatzmodellverfahren. Dieses Phänomen ist bekannt als *Fluch der Dimensionalität* (engl. *curse of dimensionality*). Ausserdem wählen wir einen rein datenbasierten Ansatz, d.h., die gesamte Analyse wird basierend auf einer begrenzten Anzahl von Beobachtungen und wenigen bis gar keinen Annahmen über die innere Funktionsweise des Systems ausgeführt. Dieses Szenario ist von hoher praktischer Relevanz, z.B. wenn für die Simulation eines Systems ein komplexer Arbeitsablauf mit mehreren Softwareprogrammen nötig ist, oder

wenn in praktischen Anwendungen nur Messungen der Eingabeparameter und der zugehörigen Modellantworten verfügbar sind. Solche datenbasierten Ansätze bringen jedoch zusätzliche Herausforderungen mit sich, die mit den (unbekannten) Eigenschaften des Eingaberaums zusammenhängen. Um diese zu quantifizieren, wird üblicherweise auf bekannte Inferenzmethoden zurückgegriffen (siehe Kapitel 2), welche allerdings auch vom Fluch der Dimensionalität betroffen sind.

Um die datenbasierte Ungewissheitsquantifizierung in hochdimensionalen Eingaberäumen zu ermöglichen, schlagen wir eine Kombination von Verfahren des maschinellen Lernens für Datenkompression und von modernsten Ersatzmodellverfahren der Ungewissheitsquantifizierung vor. Der erste wesentliche Bestandteil unseres Ansatzes, Dimensionsreduktion, wird in Kapitel 3 behandelt. Durch eine Literaturübersicht des recht umfangreichen Feldes der Dimensionsreduktion zeigen wir die Stärken und Schwächen sowie die Anwendungsbereiche verschiedener Methoden auf.

Die zweite wesentliche Zutat, Ersatzmodellierung, wird in Kapitel 4 behandelt. Über die allgemeine Formulierung hinaus liegt hier der Fokus auf den zwei Verfahren Kriging und sogenannten ›Polynomial Chaos Expansions‹, die in dieser Arbeit verwendet werden.

Eine neue Methodik für Ersatzmodellierung in hochdimensionalen Räumen wird in Kapitel 5 eingeführt. Der vorgeschlagene Algorithmus kombiniert die Kompression der Eingabedaten und die Ersatzmodellierung auf eine Weise, die die Leistung des resultierenden Ersatzmodells optimiert. Ausserdem demonstrieren wir, dass unser Algorithmus verglichen mit herkömmlichen Verfahren, welche Dimensionsreduktion und Ersatzmodellierung als zwei separate Schritte ansehen, auf mehreren Benchmarkanwendungen eine durchweg überlegene Leistung zeigt (bezüglich der Vorhersagegenauigkeit des resultierenden Ersatzmodells).

In Kapitel 6 schlagen wir einen Arbeitsablauf für datenbasierte Ungewissheitsquantifizierung in hochdimensionalen Räumen vor, welcher den Kern dieser Arbeit darstellt. Dieser Arbeitsablauf baut auf den Ergebnissen der vorhergehenden Kapitel auf. Die Verfügbarkeit eines komprimierten Raums von überschaubarer Grösse und eines Ersatzmodells erlaubt uns die Berechnung statistischer Eigenschaften der Zielgrössen, wie z.B. ihrer Momente, Quantile und sogar ihrer kompletten Wahrscheinlichkeitsverteilungen. Wir wenden diese Methodik auf Benchmarkanwendungen an und weisen nach,

dass sie besonders in den Extremwertregionen zu verbesserten Abschätzungen der Zielgrössen führen kann.

Schliesslich zeigen wir in Kapitel 7, wie die Methoden, die in dieser Arbeit vorgestellt werden, auf ein realistisches ingenieurwissenschaftliches Problem, nämlich auf die Überwachung des strukturellen Zustandes von Windkraftanlagen, angewendet werden können. Das Ziel ist die Abschätzung von Ermüdungsakkumulation und Spitzenbelastungen sowie deren Ungewissheiten für verschiedene Komponenten der Windkraftanlage basierend auf Messungen der Windgeschwindigkeit in Zeitabschnitten von jeweils 10 Minuten. Dazu verarbeiten wir eine begrenzte Anzahl an Beobachtungen, die von spezialisierter Software generiert werden.

Diese Dissertation führt neue Verfahren ein, welche die Ungewissheitsquantifizierung für eine umfangreiche Klasse von Problemen ermöglichen, für die dies vorher nicht möglich war. Für die Praxis ist dies hochrelevant, da viele heute relevante Probleme in Bereichen wie der Überwachung von strukturellen Zuständen von Bauwerken, des Erdbebensicheren Bauens, der Wettervorhersage, der Hydrogeologie und der Regelungstechnik einen hochdimensionalen Eingaberaum (beispielsweise von Zeitreihen oder Bildern) aufweisen. Die neue Methodik und unsere Erkenntnisse sowie Vorschläge für zukünftige Forschung zu diesem Thema werden in Kapitel 8 zusammengefasst.

ACKNOWLEDGEMENTS

Firstly, I would like to express my gratitude towards Prof. Sudret for his continued support and encouragement throughout our collaboration. I was always inspired by his expertise in uncertainty quantification and his analytical thinking, coupled with his leadership skills. He consistently maintained a high-productivity group while ensuring that all members are well-integrated in a pleasant working environment.

Furthermore, I would like to express my gratitude towards Dr Marelli for being my mentor over the past six years. Even after all this time working together I am still impressed by his excellence in various topics such as information technology, mathematics, machine learning and uncertainty quantification. Furthermore, I am grateful for his continuous encouragement and coaching throughout the years that gave me no room to not improve. Due to all the above, being the first PhD student graduating under his (co-) supervision, gives me a special sense of pride.

I would also like to thank my co-examiners Prof. Chatzi and Prof. Bourinet for their contributions towards the improvement of this manuscript and for pointing out research directions beyond the scope of this work.

My sincere gratitude also goes to the various Chair members that I had the pleasure to work and hangout with during my stay at the Chair. This includes the "old guard" Carlos, Roland, Joseph and Chu as well as the newer members, Paul, Philippe, Damar, Moustapha, Emiliano, Katerina and Xujia. Furthermore I would like to thank Nora Lüthen for her courage in providing me with endless amounts of wind turbine simulations and our pleasant discussions, Dr Imad Abdallah and Dr René Slot for sharing their expert knowledge on wind turbines and Margaretha Neuhaus for her continuous administrative and personal support.

Last but not least, I would like to thank my extended family of friends and relatives for their continuous support that gave me the strength to reach where I could not.

CONTENTS

1

INTRODUCTION

1.1 CONTEXT

Mathematical models lie at the foundation of our scientific understanding of the world. Scientists who try to understand a physical phenomenon, use models to describe it through an idealised representation that is expressed by a set of equations that are then validated through experimental observations. Historically, such models have helped scientists understand and formulate the fundamental laws of physics (*e.g.* gravitation, optics, electromagnetism, etc.).

In the 20th century, digital computers led to the emergence of numerical simulations, that in turn allow researchers and practitioners to describe physical phenomena with higher-fidelity models, by solving more complex underlying equations. Over the past decades, in particular, there has been an exponential increase in the computational and storage capacity of computers due to the exponential increase of the transistor counts in microprocessors (Koomey et al., 2011). Consequently, computer simulations have become nowadays an irreplaceable tool for designing engineering systems (*e.g.* vehicles, bridges, nuclear power plants, etc.), monitoring them (*e.g.* structural health monitoring, decision making in autonomous vehicles, etc.) and making predictions (*e.g.* meteorology, stock market, etc.) with ever increasing accuracy.

However, regardless of its level of fidelity, a computational model unavoidably represents an approximation of the underlying real-world phenomenon. This can be attributed to different sources of inaccuracy, depending on the application: (i) there exists no closed-form solution of the underlying problem leading to approximate solvers (*e.g.* finite element solvers) that introduce numerical and discretisation errors, (ii) not all parameters affecting the underlying process are taken into account (*e.g.* over-simplification of the underlying physics), (iii) the values of the model

parameters may not be known perfectly (*e.g.* incomplete information on the boundary conditions). Such inaccuracies in engineering models may lead to poor designs, or predictions that can potentially lead to serious consequences in terms of casualties or financial losses. To mitigate this risk during the design process of engineering systems, it is therefore critical to take into account the inaccuracies of the computational model as well as the uncertainty on its input parameters. Indeed, this has received much attention in the last two decades, thus giving rise to a new research field called uncertainty quantification. Starting from a realistic computational model that reproduces the behaviour of the system under consideration, uncertainty quantification aims at modelling the various sources of uncertainty that affect its input parameters, as well as propagating these uncertainties to the response quantities of interest (*e.g.* performance indicators).

The aforementioned radical increase in computing and storage capacity gave also rise to another trend: starting from the 90's a shift is observed in the interest of some communities (*e.g.* machine learning) from knowledge-driven to *data-driven* methodologies for constructing models (Vapnik, 1998). Such approximate models are inferred in a non-intrusive fashion, *i.e.* based only on the available data without assuming any prior knowledge on the inner workings of the system. Data-driven approaches have become increasingly popular until today due to their flexibility and the large volumes of data that are at our disposal, which may describe highly complex relationships that cannot be expressed in terms of fundamental principles from physics and maths (e.g. face/handwriting recognition, sentiment analysis and natural language understanding in general, etc.). From a historical perspective, it is worth pointing out that the recent success of such methods within the machine learning community can be attributed more to the radical increase in the computing and storage capacity and less to the novelty of the learning algorithms (Krizhevsky et al., 2012; Goodfellow et al., 2016).

Together with the increased volume of data their detail and resolution also increased *e.g.* due to more accurate sensors combined with high bandwidth transmission and storage of their readings. To put it in a quantitative perspective, the maximum dimension of a dataset in the UCI repository (Dheeru and Karra Taniskidou, 2017) has evolved from 102 in the 90's to 481 in the 2000's and approximately 3.2 million in the 2010's. The common occurrence of high dimensional data in modern engineering applications

poses challenges but also opportunities to understand and predict the behaviour of highly complex systems. The ingredients to adequately handle such problems are available. The machine learning community has devised techniques to effectively compress and model such systems but such techniques typically do not take into account uncertainties. The latter is not true for uncertainty quantification techniques that provide robust methodologies to model such systems and handle uncertainty but suffer from high dimensionality. This thesis explores ways to bridge this gap between the two communities and enable uncertainty quantification in high-dimensional data-driven problems.

1.2 GENERAL FRAMEWORK FOR UNCERTAINTY QUANTIFICATION

Uncertainty quantification (UQ) aims at taking into account the uncertainties in the parameters of the model of a physical system and at studying their impact onto the system response. It is an inter-disciplinary field that lies at the intersection between physics (*e.g.* civil and mechanical engineering) and applied mathematics (*e.g.* statistics, probability theory, computer simulation).

A well-established approach for the representation and solution of an uncertainty quantification problem is presented next (de Rocquigny, 2006a,b; Sudret, 2007). The rationale is that any such problem can be represented by a combination of four elements (steps), as sketched in Figure 1.1:

- **Step A**: definition of the computational model of the physical system. This is a rather broad step that may refer to an analytical function in its simplest form, or an entire workflow that may contain various levels of computer simulations and even physical experiments. In general, the computational model maps a set of input parameters to one or more quantities of interest (QoI), often referred to as *model responses* (or outputs).

- **Step B**: quantification of the sources of uncertainty. This step entails the identification of the input parameters that are uncertain and their description within a probabilistic context. A variety of modelling choices is available, including probability distributions, random fields, intervals and imprecise probabilities.

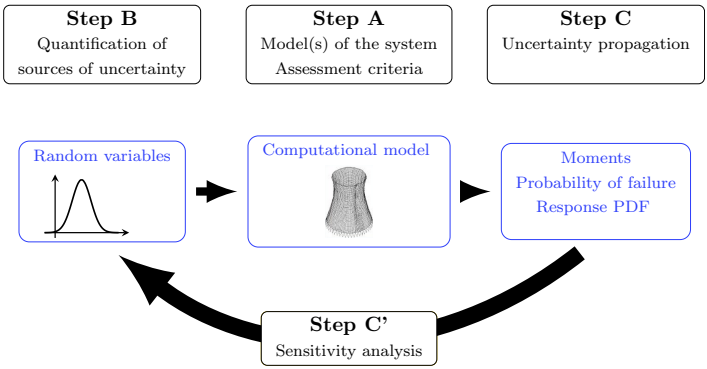


FIGURE 1.1: The framework considered for dealing with uncertainty quantification problems.

- **Step C**: uncertainty propagation. This step refers to the quantification of the uncertainty in the QoI by propagating the uncertainty of the input parameters through the computational model. Depending on the question at hand, this step may refer to the computation of various metrics such as the mean or higher-order moments, failure probability, quantiles or even the full probability distribution of the QoI.

- **Step C'**: iterative updating of the sources of uncertainty. This step may refer to several techniques used to update the information available on the sources of uncertainty identified in Step B after uncertainty propagation. Examples include *sensitivity analysis*, which can quantify the contribution of each input variable to the model response, or *Bayesian inference*, which can reduce the uncertainty in the input parameters when experimental observations are available.

One important consideration is that propagating the uncertainties (Step C) may require thousands or even millions of runs of the computational model (see *e.g.* Monte Carlo methods, Ripley, 2009). This can lead to computationally intractable problems, especially considering that modern high-fidelity models may require several hours to execute for a single set of input parameters. A powerful class of techniques used in the recent literature to solve this problem is based on computationally inexpensive *surrogate models* (Sudret, 2007; Eldred et al., 2016). A surrogate model (also known as *metamodel*) is an analytical function that provides an accurate approximation of

a computational model, based on a limited number of runs of the simulator at selected values of the input parameters (the so-called *experimental design*) and some learning algorithm. An additional benefit of using surrogate models is that they are often non-intrusive, *i.e.* their construction only depends on the experimental design without requiring access to the model itself.

1.3 PROBLEM STATEMENT

Modern engineering applications are considered that simulate complex processes depending on a large number of input parameters. Typical examples are simulators that depend on spatial/temporal inputs. As an example, high-fidelity wind-turbine simulators take time series of simulated wind speed as inputs in order to predict QoI's such as loads on substructures (*e.g.* blades) and fatigue. Furthermore, the computational model is often a black box, *i.e.* only the input parameters and the model responses are available. The internal workings of the computational model are not accessible. This can be due, *e.g.* to complex workflows involving various software packages to simulate a system or real-world applications where only measurements of the input parameters and model responses are available. The focus of this thesis is such data-driven applications with high-dimensional inputs. Dealing with uncertainties in such systems introduces multiple challenges.

When the joint probability distribution of the input variables is unknown, as in data-driven applications, it can sometimes be estimated by fitting complex probabilistic models (Torre et al., 2018). Although there are well-established approaches for fitting distributions to existing data, this task becomes non-trivial when the number of input variables is large (i.e. $\mathcal{O}(10^2 - 10^4)$). In addition, the number of unknown distribution parameters can be too large to be inferred from the limited amount of samples in the available data set (leading to an under-determined problem).

Even in the case where an adequate probabilistic input model is available, obtaining the corresponding model responses can pose a major challenge, given that the true model is either not available or is computationally too expensive to directly perform Monte Carlo simulation. In such cases, the underlying model is substituted by a surrogate. In high dimension, however, the performance of surrogate models decreases, while the cost of computing

and storing them increases. This is a well-known issue known as the *curse of dimensionality* (Verleysen and François, 2005). The surrogate computation may even be intractable when the number of input parameters is $\mathcal{O}(10^3)$ or larger.

1.4 OBJECTIVES AND OUTLINE OF THE THESIS

The goal of this Ph.D. thesis is to propose new algorithms that enable data-driven uncertainty quantification for engineering systems characterised by a large number of input variables. This goal is detailed by the following objectives:

(i) Enable surrogate modelling in problems with high-dimensional inputs

(ii) Use state-of-the art surrogate modelling and dimensionality reduction techniques

(iii) Apply the proposed approach to a variety of benchmark problems

(iv) Show the relevance of the proposed methodology for realistic engineering problems

In order to address the objectives, the thesis is organised in eight chapters including this introduction.

Chapter 2 introduces the standard methodology for quantifying the sources of uncertainty. In particular, the notion of random variables and random vectors is introduced. For random vectors, the general case of dependence is addressed using the copula formalism.

To deal with high-dimensional surrogate modelling, dimensionality reduction is explored in Chapter 3. It contains a comprehensive review of state-of-the-art dimensionality reduction techniques. Starting from linear mappings of the high-dimensional space to a lower-dimensional one, additional focus is given to their non-linear extensions. These transformations have been successfully used in applications related to supervised learning, visualisation and image de-noising.

The second ingredient for high-dimensional surrogate modelling, is discussed in Chapter 4, which focuses on classical surrogate modelling techniques. Two state-of-the-art methods are discussed in detail, namely Gaus-

sian process modelling (also known as Kriging) and polynomial chaos expansions. Both methods are suitable for data-driven applications but suffer from the curse of dimensionality.

A novel algorithm for performing high-dimensional surrogate modelling is proposed in Chapter 5. It is based on a non-intrusive combination of the previously presented ingredients (dimensionality reduction and surrogate modelling) in a way that the compressed input space achieves optimal surrogate model performance. The performance of the proposed methodology is compared against traditional approaches that sequentially perform dimensionality reduction and surrogate modelling on a set of benchmark applications.

Chapter 6 capitalises on the results of Chapter 5 and showcases the full framework for high-dimensional, data-driven uncertainty propagation. Having access to a compressed input space and a surrogate model allows one to first fit a probabilistic model, in the compressed input space, and then perform forward uncertainty propagation using standard techniques.

In Chapter 7 the proposed framework is applied to a realistic engineering problem. The system under consideration is a model of a wind turbine. The input parameters of the model are time series of realistic wind speeds that were generated using the TurbSim software (Jonkman, 2009). The wind speed time series are fed to an aero-servo-elastic simulator, called OpenFAST (Jonkman, 2013), that calculates the QoI corresponding to fatigue damage of various components of the wind turbine. Based on a limited set of available simulation samples, the proposed methodology is used to compute a surrogate model on a compressed input space that predicts the fatigue loads given a new wind-speed-over-time recording.

Chapter 8 finally provides an overview of the new methodology and results obtained during the Ph.D. work as well as suggestions for future research on the topic of data-driven uncertainty quantification in high dimension.

1.5 PUBLICATIONS

Articles in peer-reviewed journals:

- **Lataniotis, C.**, S. Marelli and B. Sudret (2019). Extending classical surrogate modelling to ultrahigh dimensional problems through su-

pervised dimensionality reduction: a data-driven approach. *Int. J. Uncertainty Quantification*. Submitted

- Abdallah, I., **C. Lataniotis**, and B. Sudret (2019). Parametric hierarchical Kriging for multi-fidelity aero-servo-elastic simulators - application to extreme loads on wind turbines. *Prob. Eng. Mech. 55*, 67–77

- **Lataniotis, C.**, S. Marelli and B. Sudret (2018). The Gaussian process modelling module in UQLab. *Soft Comput. Civil Eng. 2(3)*, 91–116

Conference contributions (with proceedings):

- Wagner, P. M., S. Marelli, **C. Lataniotis**, and B. Sudret (2019). Sequential piecewise PCE approximation of likelihood functions in Bayesian inference. In *Proc. 13th Int. Conf. on Applications of Stat. and Prob. in Civil Engineering (ICASP13), Seoul, S. Korea*. Talk given by P. Wagner.

- Abdallah, I., B. Sudret, **C. Lataniotis**, J. Sørensen, and A. Natarajan (2015). Fusing simulation results from multifidelity aero-servo-elastic simulators - Application to extreme loads on wind turbine. In *Proc. 12th Int. Conf. on Applications of Stat. and Prob. in Civil Engineering (ICASP12), Vancouver, Canada*. Talk given by I. Abdallah.

Conference contributions (without proceedings):

- **Lataniotis, C.**, S. Marelli, and B. Sudret (2019). Data-driven uncertainty propagation in high-dimension. In *Proc. 3rd Int. Conf. Uncertainty Quantification in Computational Sciences and Engineering (UNCECOMP), Crete, Greece*. Talk given by S. Marelli.

- **Lataniotis, C.**, S. Marelli, and B. Sudret (2018). Dimensionality reduction and surrogate modelling for high-dimensional UQ problems. In *Workshop on "Reducing dimensions and cost for UQ in complex systems", Isaac Newton Institute for Mathematical Sciences, Cambridge, United Kingdom*. Invited lecture given by B. Sudret.

- **Lataniotis, C.**, S. Marelli, and B. Sudret (2017). Dimensionality reduction and surrogate modelling for high-dimensional UQ problems. In *MascotNum Annual Conference, Paris-Massy, France*. Talk given by C. Lataniotis.

- **Lataniotis, C.**, S. Marelli, and B. Sudret (2017). Dimensionality reduction and surrogate modelling for high-dimensional UQ problems. In *Proc. 2nd Int. Conf. Uncertainty Quantification in Computational Sciences and Engineering (UNCECOMP), Rhodes Island, Greece*. Talk given by C. Lataniotis.

- **Lataniotis, C.**, S. Marelli, and B. Sudret (2016). Combining feature mapping and Gaussian process modelling in the context of uncertainty quantification. In *SIAM Conf. on Uncertainty Quantification, Lausanne, Switzerland*. Talk given by C. Lataniotis.

- Marelli, S., **C. Lataniotis**, and B. Sudret (2016). Advancements in the UQLab framework for uncertainty quantification. In *SIAM Conf. on Uncertainty Quantification, Lausanne, Switzerland*. Talk given by S. Marelli.

- Marelli, S., B. Sudret, **C. Lataniotis**, and R. Schöbi (2016). UQLab: A general-purpose Matlab-based platform for uncertainty quantification. In *SIAM Conf. on Uncertainty Quantification, Lausanne, Switzerland*. Talk given by S. Marelli.

- Sudret, B., S. Marelli, and **C. Lataniotis** (2015). Sparse polynomial chaos expansions as a machine learning regression technique. In *International Symposium on Big Data and Predictive Computational Modeling, Munich, Germany*. Talk given by B. Sudret.

2

PROBABILISTIC INPUT MODELS

2.1 INTRODUCTION

The identification and modelling of the sources of uncertainty of a system is a crucial step within any uncertainty quantification workflow. In a probabilistic setting, the uncertain model parameters are represented by random variables. Having multiple uncertain parameters, which may present a non-trivial dependence structure, leads to the introduction of random vectors. Within a data-driven context, no prior knowledge is assumed about those representations. Therefore, they need to be inferred from a limited number of observations. These fundamental concepts from probability theory are briefly discussed in the following sections, focusing on the formalism and the tools that will be used in later chapters of the thesis.

2.2 RANDOM VARIABLES

2.2.1 *Definitions*

Random variables are variables whose values depend on the outcome of random phenomena. More formally, a random variable X is a mapping $X : \Omega \to \mathcal{D}_X$, where Ω is the set of all possible outcomes of a phenomenon and $\mathcal{D}_X \subseteq \mathbb{R}$ the domain of X. When $\mathcal{D}_X$ is a discrete (resp. continuous) set, the variable is called discrete (resp. continuous). The cumulative distribution function (CDF) of a random variable, denoted by $F_X(x)$ is defined as:

$$F_X(x) \stackrel{\text{def}}{=} \mathbb{P}\left(X \leq x\right). \tag{2.1}$$

It is a monotonically increasing function bounded in the $[0, 1]$ interval that completely defines the random variable.

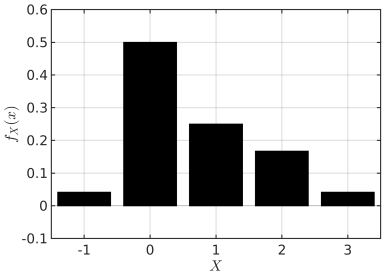
Another function that is commonly used in probability theory is related to the probability of X taking a specific value, in the discrete case, or X

falling within an infinitely small range of values in the continuous case. For discrete random variables, the probability mass function is defined as:

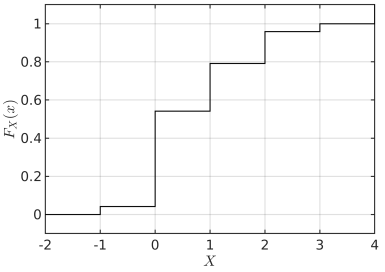
$$p_i \stackrel{\text{def}}{=} \mathbb{P}\left(X = x^{(i)}\right), \tag{2.2}$$

for a discrete domain $\mathcal{D}_X = \left\{x^{(i)}, i \in \mathbb{N}\right\}$. For continuous random variables, the probability density function (PDF) is defined as:

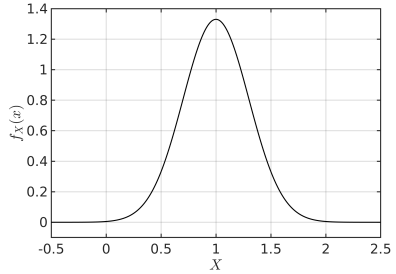
$$f_X(x) \stackrel{\text{def}}{=} \lim_{h \to 0,\ h>0} \frac{\mathbb{P}\left(x \leq X \leq x + h\right)}{h} \tag{2.3}$$



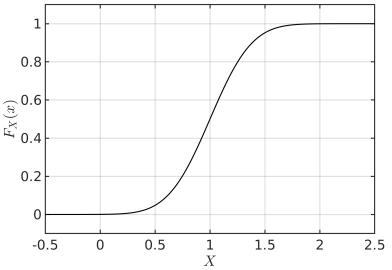
(a) Probability mass function (discrete r.v.)



(b) Cumulative density function (discrete r.v.)



(c) Probability density function (continuous r.v.)



(d) Cumulative density function (continuous r.v.)

FIGURE 2.1: Examples of probability measures for discrete and continuous random variables.

The CDF is directly connected to the probability mass/density function of a random variable. In the discrete case, the CDF reads:

$$F_X(x) = \sum_{i=1}^{N} p_i \, \mathbf{1}_{\left\{x \geq x^{(i)}\right\}}(x), \tag{2.4}$$

where $\mathbf{1}_{\{x \geq \alpha\}}$ is the indicator function, defined by:

$$\mathbf{1}_{\{x \geq \alpha\}} \overset{\text{def}}{=} \begin{cases} 1 & \text{, if } x \geq \alpha \\ 0 & \text{, otherwise} \end{cases}. \tag{2.5}$$

In the continuous case, the CDF reads:

$$F_X(x) = \int_{-\infty}^{x} f_X(x)dx \Leftrightarrow f_X(x) = \frac{dF_X(x)}{dx}. \tag{2.6}$$

In Figure 2.1 a number of visualisations of the aforementioned probability measures is given. The first row corresponds to an example of a discrete random variable and its probability mass function (Figure 2.1a) and CDF visualisation (Figure 2.1b). The second row corresponds to similar plots (PDF in Figure 2.1c and CDF in Figure 2.1d) of a continuous random variable instead.

The statistical moments of a function (in this case the PDF of X) are often used to summarise the statistics of X, by providing quantitative measures of the shape of $f_X(x)$. The n-th moment (resp. centred moment) of X is defined as:

$$\mathbb{E}\left[X^n\right] \overset{\text{def}}{=} \int_{\mathcal{D}_X} x^n f_X(x)dx, \tag{2.7}$$

$$\mathbb{E}\left[(X - \mu_X)^n\right] \overset{\text{def}}{=} \int_{\mathcal{D}_X} (x - \mu_X)^n f_X(x)dx, \tag{2.8}$$

where

$$\mu_X = \mathbb{E}\left[X\right] = \int_{\mathcal{D}_X} x f_X(x)dx \tag{2.9}$$

is the mean or expected value of X, that corresponds to the 1st moment. Other commonly used measures are the variance, standard deviation and coefficient of variation of X, that are defined as follows:

$$\mathrm{Var}[X] = \mathbb{E}\left[(X - \mu_X)^2\right], \tag{2.10}$$

$$\sigma_X = \sqrt{\mathrm{Var}[X]}, \tag{2.11}$$

$$CV_X = \frac{\sigma_X}{\mu_X}. \tag{2.12}$$

The covariance of two random variables X, Y provides a measure of the joint variability between them and is defined as:

$$\mathrm{Cov}[X, Y] = \mathbb{E}\left[(X - \mu_X)(Y - \mu_Y)\right]. \tag{2.13}$$

The magnitude of the covariance is not easy to interpret because it depends on the variance of each random variable. Hence, a normalised version can be used instead, called the *Pearson correlation coefficient*:

$$\rho = \frac{\mathrm{Cov}[X, Y]}{\sigma_X \, \sigma_Y}, \tag{2.14}$$

that is bounded in $[-1, 1]$.

2.2.2 *Univariate distribution example*

For continuous random variables (the focus in this thesis), numerous distributions are commonly used in engineering applications. One such example is given by Gaussian (or normal) distributions, denoted by $X \sim \mathcal{N}(\mu, \sigma^2)$. They are fully characterised by two parameters, namely their mean, μ, and standard deviation, σ (or equivalently the variance σ^2).

The standard normal distribution is defined as a Gaussian distribution with $\mu = 0$ and $\sigma = 1$. Its PDF reads:

$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}, \tag{2.15}$$

and its CDF is implicitly given by (based on Eq. (2.6)):

$$\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx. \tag{2.16}$$

The PDF and CDF of any Gaussian random variable $X \sim \mathcal{N}\left(\mu, \sigma^2\right)$ can be calculated by:

$$f_X(x) = \frac{1}{\sigma}\, \phi\left(\frac{x-\mu}{\sigma}\right), \tag{2.17}$$

and

$$F_X(x) = \Phi\left(\frac{x-\mu}{\sigma}\right), \tag{2.18}$$

respectively.

2.3 RANDOM VECTORS

2.3.1 *Definitions*

A real-valued random vector $\mathbf{X}$ refers to a mapping $\mathbf{X} : \Omega \to \mathcal{D}_{\mathbf{X}} \subseteq \mathbb{R}^M$, where M is the size of the vector. Random vectors correspond to a collection of M random variables, *i.e.* $\mathbf{X} = \{X_1, \ldots, X_M\}$. They are completely specified by their joint CDF that reads:

$$F_{\mathbf{X}}(\mathbf{x}) = \mathbb{P}\left(X_1 \leq x_1, \ldots, X_n \leq x_n\right). \tag{2.19}$$

Similarly to Eq. (2.6), the joint PDF of $\mathbf{X}$ reads:

$$f_{\mathbf{X}}(\mathbf{x}) = \frac{\partial^M F_{\mathbf{X}}(\mathbf{x})}{\partial x_1 \cdot \ldots \cdot \partial x_M}. \tag{2.20}$$

The marginal distribution of a component X_i is obtained by integrating the joint PDF over the remaining components:

$$f_{X_i}(x_i) = \int_{\mathcal{D}_{\mathbf{X}}^{\sim i}} f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x}_{\sim i}, \tag{2.21}$$

where $\mathcal{D}_{\mathbf{X}}^{\sim i}$ is the subset of $\mathcal{D}_{\mathbf{X}}$ where x_i is fixed and $d\mathbf{x}_{\sim i} = dx_1 \cdots dx_{i-1} dx_{i+1} \cdot \cdots dx_M$.

When dealing with random vectors, the concept of dependency plays a critical role. The components of $\mathbf{X} = \{X_1, \ldots, X_M\}$ are considered mutually independent if and only if:

$$F_{\mathbf{X}}(\mathbf{x}) = \prod_{i=1}^{M} F_{X_i}(x_i), \tag{2.22}$$

and equivalently, $f_{\mathbf{X}}(\mathbf{x}) = \prod_{i=1}^{M} f_{X_i}(x_i)$.

2.3.2 *Multivariate distribution example*

Gaussian random vectors are commonly used in the engineering practice. They are the multivariate extension of Gaussian random variables that were introduced in the previous section. A Gaussian random vector denoted by $\mathbf{X} \sim \mathcal{N}(\mu_{\mathbf{X}}, \boldsymbol{C})$ is completely defined by its mean value vector $\mu_{\mathbf{X}}$ and its covariance matrix $\boldsymbol{C}$ though the following joint PDF:

$$f_{\mathbf{X}}(\mathbf{x}) = (2\pi)^{-M/2} (\det \boldsymbol{C})^{-1/2} \exp\left[-\frac{1}{2} (\mathbf{x} - \mu_{\mathbf{X}})^{\top} \boldsymbol{C}^{-1} (\mathbf{x} - \mu_{\mathbf{X}})\right]. \tag{2.23}$$

The mean vector of $\mathbf{X}$ contains the expected value of each component, *i.e.* $\mu_{\mathbf{X}} = \{\mu_{X_1}, \ldots, \mu_{X_M}\}$. The covariance matrix of $\mathbf{X}$ is a square, symmetric and positive definite matrix of size $M \times M$ with elements:

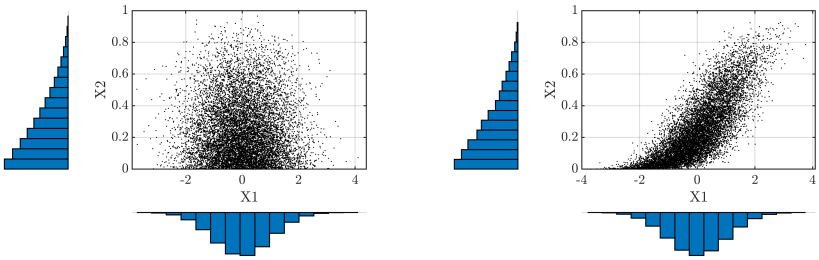
$$C_{i,j} = \mathrm{Cov}\left[X_i, X_j\right]. \tag{2.24}$$

2.3.3 *Copulas*

To deal with dependency in a structured manner, we adopt here the copula formalism (Nelsen, 2006). At the basis of this approach lies Sklar's theorem (Sklar, 1959) which states that for any multivariate CDF $F_{\mathbf{X}}$ there exists a copula C such that:

$$F_{\mathbf{X}}(\mathbf{x}) = C\left(F_{X_1}(x_1), \ldots, F_{X_M}(x_M)\right), \tag{2.25}$$

where $F_{X_i}(x_i)$ denotes the marginal CDF of X_i. In probabilistic terms, C is a joint cumulative distribution of an M-dimensional random vector



(a) Independence copula

(b) Gaussian copula

FIGURE 2.2: Comparison between samples of 2D random vectors with identical marginal distributions and different copulas.

on the unit cube $[0,1]^M$ with uniform marginals. Therefore, the copula C provides a link between the joint and marginal CDF's. Moreover, it does not depend on the marginal distributions and purely describes the statistical interactions among the components of $\mathbf{X}$. The copula formalism is particularly appealing in engineering applications because it allows one to reason with marginal distributions and dependence separately, each having a clear physical meaning.

The joint PDF of $\mathbf{X}$ can be expressed using the copula formalism analogously to Eq. (2.25) as follows:

$$f_{\mathbf{X}}(\mathbf{x}) = c\left(F_{X_1}(x_1), \ldots, F_{X_M}(x_M)\right) \prod_{i=1}^{M} f_{X_i}(x_i), \tag{2.26}$$

where $c(\cdot)$ is the copula density function that is defined as:

$$c(u_1, \ldots, u_M) \stackrel{\text{def}}{=} \frac{\partial^M C(u_1, \ldots, u_M)}{\partial u_1 \cdots \partial u_M}. \tag{2.27}$$

The case of mutual independence among the random variables can be modelled using the *independent copula*, often denoted as $C^{\perp}$, which reads:

$$C^{\perp}(\boldsymbol{u}) = \prod_{i=1}^{M} u_i \,. \tag{2.28}$$

According to this copula definition, it is clear that independence as specified in Eq. (2.22) follows from Eq. (2.25) and Eq. (2.28).

Arguably the best-known copula in engineering (Nataf, 1962; Lebrun and Dutfoy, 2009) is the *Gaussian copula* that reads:

$$C(\boldsymbol{u}) = \Phi_{\boldsymbol{R}}\left(\Phi^{-1}(u_1), \dots, \Phi^{-1}(u_M)\right), \tag{2.29}$$

where $\Phi_{\boldsymbol{R}}$ denotes the joint CDF of a multivariate Gaussian distribution with zero mean and covariance matrix equal to the correlation matrix $\boldsymbol{R}$ that reads:

$$\boldsymbol{R} = \left[\rho_{ij}\right] \;\; i, j = 1, \dots, M, \tag{2.30}$$

and Φ^{-1} denotes the univariate standard normal inverse CDF. It can be shown that a random vector specified by a Gaussian copula with correlation matrix $\boldsymbol{R}$ and Gaussian marginal distributions with means $\mu_1, \dots, \mu_M$ and standard deviations $\sigma_1, \dots, \sigma_M$ is equivalent to a multivariate Gaussian random vector with mean vector $\boldsymbol{\mu} = [\mu_1, \dots, \mu_M]$ and covariance matrix with elements $C_{ij} = \rho_{ij}\sigma_i\sigma_j$.

A comparison between samples from a 2D random vector with identical marginal distributions and different dependency type is shown in Figure 2.2. In Figure 2.2a the independence copula is used and as a consequence the samples are evenly scattered. In Figure 2.2b a Gaussian copula is used with $\rho = 0.8$, that results in samples with strong (positive) correlation between X_1 and X_2. Notice that some extreme regions, *e.g.* $X_1, X_2 > 2$ are more densely covered in the latter case. This highlights the importance of dependence in uncertainty quantification: neglecting it (which may occur in the engineering practice) may lead to poor decisions, due to *e.g.* underestimating the probability of occurrence of an extreme event (Torre et al., 2018).

Various parametric copula families have been proposed in the literature along with techniques for combining those and achieving more elaborate dependence structures (see *e.g.* Nelsen, 2006; Joe, 2015).

2.4 DATA-DRIVEN INFERENCE

The focus of this thesis is given to data-driven approaches for estimating the joint distribution $F_{\mathbf{X}}(\mathbf{x})$ of a random variable $\mathbf{X}$, given only a dataset

$\mathcal{X} = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\right\}$ and no additional knowledge about the probabilistic model of $\mathbf{X}$.

From Eq. (2.25) it follows that the estimation of the unknown joint CDF $F_{\mathbf{X}}(\mathbf{x})$ can be decomposed into two separate problems: (i) estimation of the marginal distributions $F_{X_1}(x_1), \ldots, F_{X_M}(x_M)$, and, (ii) estimation of the copula function.

2.4.1 *Inference of the marginal distributions*

Thanks to the copula formalism, fitting each marginal distribution can be done independently from the others, hence we consider here a single random variable X without loss of generality. There exist numerous approaches for estimating the distribution of X (see *e.g.* Bishop, 2006; Ang and Tang, 2007), that can be broadly classified into two classes: (i) *parametric* techniques that assume a specific distribution family and then estimate the most likely values of their parameters, and, (ii) *non-parametric* techniques that make few assumptions about the underlying distribution instead. The latter are typically more appealing in data-driven applications where one may not be able to assume one specific distribution over some other a priori. A non-parametric inference technique is mainly used in this work that is called *kernel density estimation* or *kernel smoothing* (Rosenblatt, 1956; Parzen, 1962).

The kernel density estimate $\widehat{f}$ of the unknown PDF f reads:

$$\widehat{f}_X(x) = \frac{1}{N\,h} \sum_{i=1}^{N} \kappa\left(\frac{x - x^{(i)}}{h}\right), \tag{2.31}$$

where $\kappa(\cdot)$ denotes the kernel function and h an unknown parameter called the *bandwidth* that is inferred from the samples in $\mathcal{X}$. This approach constructs the unknown PDF by summing N elementary kernel functions, each centred in one of the samples of $\mathcal{X}$. Different kernels may be preferred depending on the modes (peaks) and skewness of the density function. The kernel bandwidth is a free parameter that exhibits a strong influence on the resulting estimate. Its value can be determined using empirical or closed-form solutions depending on the kernel that is used. For more information on the topic of kernel and bandwidth selection see *e.g.* Silverman (2018).

2.4.2 *Copula inference*

The second part of fitting an unknown joint distribution of $\mathbf{X}$ refers to the copula inference. Recall from Eq. (2.26) that the joint PDF is expressed as:

$$f_{\mathbf{X}}(\mathbf{x};\zeta) = c\left(F_{X_1}(x_1), \ldots, F_{X_M}(x_M);\boldsymbol{\zeta}\right)\prod_{i=1}^{M} f_{X_i}(x_i), \tag{2.32}$$

where $\boldsymbol{\zeta}$ refers to the unknown copula parameters. A typical approach for inferring the values of $\boldsymbol{\zeta}$ is the *maximum-likelihood method*. The likelihood of a PDF $f_{\mathbf{X}}(\mathbf{x};\boldsymbol{\zeta})$ given $\mathcal{X}$ is defined as follows:

$$\mathcal{L}\left(\boldsymbol{\zeta}\right) = \prod_{i=1}^{N} f_{\mathbf{X}}(\mathbf{x}^{(i)};\boldsymbol{\zeta}) = \prod_{i=1}^{N} c\left(F_{X_1}(x_1^{(i)}), \ldots, F_{X_M}(x_M^{(i)});\boldsymbol{\zeta}\right)\prod_{j=1}^{M} f_{X_j}(x_j^{(i)}). \tag{2.33}$$

The goal is to find such $\widehat{\boldsymbol{\zeta}}$ that maximises the likelihood function, *i.e.*

$$\widehat{\boldsymbol{\zeta}} = \underset{\mathcal{D}_{\boldsymbol{\zeta}}}{\arg\max}\ \mathcal{L}\left(\boldsymbol{\zeta}\right), \tag{2.34}$$

or equivalently, the minimiser of negative log-likelihood:

$$\widehat{\boldsymbol{\zeta}} = \underset{\mathcal{D}_{\boldsymbol{\zeta}}}{\arg\min}\ -\log\left(\mathcal{L}\left(\boldsymbol{\zeta}\right)\right), \tag{2.35}$$

where $\mathcal{D}_{\boldsymbol{\zeta}}$ is the domain of $\boldsymbol{\zeta}$. The rationale of this method is that by using the appropriate $\boldsymbol{\zeta}$ values, more samples from $\mathcal{X}$ will fall within more likely regions (large PDF value) instead of less likely ones (small PDF value). In the general case, there is no closed-form solution of Eq. (2.34) and gradient-based or global optimisation methods are used for calculating $\widehat{\boldsymbol{\zeta}}$ instead.

3

DIMENSIONALITY REDUCTION

3.1 INTRODUCTION

Nowadays, high dimensional data arise in various applications. That is, the random variable $\mathbf{X}$ that quantifies some phenomenon lies in a space $\mathcal{D}_{\mathbf{X}} \subseteq \mathbb{R}^M$ where M is large. Learning methods, such as surrogate modelling, tend to under-perform when applied to high-dimensional spaces. Furthermore, technical constraints apply to the storage and processing of high dimensional data: as the data volume increases, compression becomes increasingly important. In order to handle such data adequately, its dimensionality needs to be reduced.

In an abstract sense, dimensionality reduction (DR) refers to the parametric mapping $g : \mathcal{D}_{\mathbf{X}} \subseteq \mathbb{R}^M \to \mathcal{D}_{\mathbf{Z}} \subseteq \mathbb{R}^m$ of the form:

$$\mathbf{Z} = g(\mathbf{X}; \mathbf{w}) \,, \tag{3.1}$$

where $\mathbf{w}$ is the set of parameters associated with the mapping. Dimensionality reduction occurs if $m << M$, *i.e.* if $m = \mathcal{O}\left(10^{0-1}\right)$ whereas $M = \mathcal{O}\left(10^{2-4}\right)$. The nature and number of the parameters $\mathbf{w}$ depends on the specific DR method under consideration.

Such transformations are motivated by the assumption that $\mathbf{X}$ lies on a manifold with dimensionality m that is embedded within the M-dimensional space. The specific value of m is in some applications referred to as the "intrinsic dimension" (ID) of $\mathbf{X}$ (Fukunaga, 2013). From an information theory perspective, the ID refers to the minimum number of scalar variables that is required to represent $\mathbf{X}$ without any loss w.r.t. an appropriate information measure. In practice, the ID of $\mathbf{X}$ is often unknown. In such cases, m becomes a parameter of dimensionality reduction, and it can either be imposed, or inferred from available data by various approaches (see *e.g.* Camastra (2003) for a comparative overview).

A common aspect of all parametric DR methods is that for each choice of the ID m, the remaining parameters $\mathbf{w}$ are estimated by minimising a suitable error measure (also called *loss function*):

$$\widehat{\mathbf{w}} = \arg\min_{\mathcal{D}_{\mathbf{w}}} J(\mathbf{w}), \tag{3.2}$$

where $\widehat{\mathbf{w}}$ denotes the parameter estimate, $\mathcal{D}_{\mathbf{w}}$ the feasible domain of $\mathbf{w}$ and $J(\cdot)$ the error measure. The choice of error measure depends on the specific application DR is used for. When the goal is direct compression of a high dimensional input without information loss (a common situation in image compression, telecommunications, etc.), a common choice of $J(\cdot)$ is the so-called mean-squared reconstruction error, that reads:

$$J(\mathbf{w}) = \mathbb{E}\left[\left\|\mathbf{X} - \widehat{\mathbf{X}}\right\|^2\right], \tag{3.3}$$

where $\widehat{\mathbf{X}} = g^{-1}(\mathbf{Z}, \mathbf{w}')$ denotes the reconstruction of $\mathbf{X}$, calculated through the inverse transform $g^{-1} : \mathcal{D}_{\mathbf{Z}} \to \mathcal{D}_{\mathbf{X}}$. The parameters of the inverse transform are denoted as $\mathbf{w}'$ to highlight that in general they are different from those of the forward transform g. Depending on the DR method, the inverse transform may not exist and one can only settle for an approximate solution (see *e.g.* Kwok and Tsang (2003)).

The focus in this thesis is data-driven dimensionality reduction. In such cases the values of the parameters $\mathbf{w}$ are inferred based on the available data (also known as experimental design) $\mathcal{X} = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\right\}$, *i.e.* Eq. (3.2) is modified as follows:

$$\widehat{\mathbf{w}} = \arg\min_{\mathcal{D}_{\mathbf{w}}} J(\mathbf{w}; \mathcal{X}). \tag{3.4}$$

Hence, considering for example the loss function in Eq. (3.3), its value is approximated as follows within a data-driven context:

$$J(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{x}^{(i)} - \widehat{\mathbf{x}}^{(i)}\right\|^2. \tag{3.5}$$

A rich literature about dimensionality reduction techniques is available (see *e.g.* Van Der Maaten et al. (2009)). This chapter describes in more detail a selection of such techniques that have received wide attention within the

machine learning community over the past decades. The review starts with principal component analysis (PCA) in Section 3.2, a DR technique that was conceived towards the end of the 19$^{\text{th}}$ century and is still prevalent today. PCA is linear which may result in limited compression capabilities in realistic applications. Hence, non-linear techniques are introduced next, starting from multidimensional scaling (MDS) in Section 3.3. The classical MDS has limited application today but laid the foundation for widely popular techniques such as Isomap which are also described. Kernel PCA is presented next in Section 3.4. The importance of this technique lies in the flexibility to elegantly perform potentially highly complex transformations using the so-called kernel trick. In addition, a modern class of neural networks suitable for DR is described in Section 3.5. Finally, Section 3.6 provides a comparison between the presented methods in terms of their compressive performance on benchmark datasets.

3.2 PRINCIPAL COMPONENT ANALYSIS

Principal Component Analysis (PCA) is a dimensionality reduction technique that aims at calculating a linear basis of $\mathcal{D}_{\mathbf{X}}$ with reduced dimensionality that preserves the sample variance (Pearson, 1901). Given an experimental design $\mathcal{X}$, the PCA algorithm is based on the eigen-decomposition of the sample covariance matrix $\boldsymbol{C}$:

$$\boldsymbol{C} = \frac{1}{N} \bar{\mathcal{X}}^{\top} \bar{\mathcal{X}}, \tag{3.6}$$

of the form:

$$\boldsymbol{C}\boldsymbol{v}^{(i)} = \lambda^{(i)} \boldsymbol{v}^{(i)}\,,\ i = 1, \ldots, M\,, \tag{3.7}$$

where $\bar{\mathcal{X}}$ denotes the centred (zero mean) experimental design, $\lambda^{(i)}$ denotes each eigenvalue of $\boldsymbol{C}$ and $\boldsymbol{v}^{(i)}$ the corresponding eigenvector.

Then the compressed samples are obtained by:

$$\mathbf{z} = (\mathbf{x} - \boldsymbol{\mu}_{\mathbf{X}})\ \boldsymbol{V}\,, \tag{3.8}$$

where $\boldsymbol{\mu}_{\mathbf{X}}$ is the sample mean of $\mathbf{X}$ evaluated on $\mathcal{X}$ and $\boldsymbol{V}$ is the $M \times m$ collection of m eigenvectors of $\boldsymbol{C}$ with maximal eigenvalues. The compres-

sion is achieved due to the selection of a set of m out of M eigenvectors. Those eigenvectors are called the *principal components* of $\mathcal{X}$ because they correspond to the reduced basis of $\mathcal{X}$ with maximal variance (see *e.g.* Bishop (2006) for a proof).

Given $\mathbf{z} \in \mathbb{R}^m$, it is straightforward to obtain its reconstruction $\widehat{\mathbf{x}} \in \mathbb{R}^M$ by:

$$\widehat{\mathbf{x}} = \mathbf{z}\,\boldsymbol{V}^T + \boldsymbol{\mu}_{\mathbf{X}}. \tag{3.9}$$

For a full rank covariance matrix $\boldsymbol{C}$, the transformation in Eq. (3.9) is exact when $m = M$.

There is a close relationship between PCA and the singular value decomposition of $\mathbf{X}$ (see *e.g.* Murphy (2012)). Indeed, due to improved numerical stability, the latter approach is typically preferred for calculating the eigenvectors and eigenvalues of $\boldsymbol{C}$ in practice.

Given that standard SVD algorithms have $\mathcal{O}(N^3)$ computational complexity (Knockaert et al., 1999), PCA may be infeasible for large sample sizes N. In such cases, approximate methods can be used. Simple PCA provides an iterative algorithm for estimating the principal components (Partridge and Calvo, 1998) and other methods are approximating the singular value decomposition of $\mathbf{X}$ using random projections (Rokhlin et al., 2009; Halko et al., 2011).

Another popular extension of PCA is the so-called *probabilistic PCA* (PPCA). It is based on the assumption that m latent (*i.e.* unobserved) variables collected in $\mathbf{z} \in \mathbb{R}^m$ are associated with $\mathbf{x} \in \mathbb{R}^M$ as follows (Tipping and Bishop, 1999b):

$$\mathbf{X} = \mathbf{W}\mathbf{Z} + \boldsymbol{\mu} + \boldsymbol{\epsilon}\,, \tag{3.10}$$

where the parameter vector $\boldsymbol{\mu}$ allows this representation to have non-zero mean and $\boldsymbol{\epsilon} \sim \mathcal{N}\left(\mathbf{0}, \sigma^2 \boldsymbol{I}\right)$ is a zero-mean, multivariate Gaussian random variable that represents the error (or noise) of the latent variable model. Conventionally, the latent variables follow a standard normal distribution ($\mathbf{Z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$) leading, by virtue of Eq. (3.10), to Gaussian distributed

observations $\mathbf{X} \sim \mathcal{N}\left(\boldsymbol{\mu}, \mathbf{W}\mathbf{W}^\top + \sigma^2 \boldsymbol{I}\right)$. The latent variable model parameters can thus be determined by maximum likelihood (Tipping and Bishop, 1999b).

Such probabilistic formulation extends the capabilities of classical PCA in certain aspects. Having access to a likelihood function allows one to construct mixtures of PPCA's, that may be beneficial in more complex problems (Tipping and Bishop, 1999a). PPCA also enables the application of iterative and computationally efficient expectation-maximisation algorithms for performing PCA (Tipping and Bishop, 1999a,b). Finally it is noteworthy to mention that PPCA generalises the concept of PCA by introducing a Gaussian noise term. In fact, it can be shown that PPCA with $\sigma = 0$ is equivalent to PCA (see *e.g.* Bishop (2006)).

3.3 MULTI-DIMENSIONAL SCALING

Multidimensional scaling (MDS) is a dimensionality reduction technique that aims at minimising the dissimilarity of the samples between the initial and reduced space. Given the high-dimensional samples $\mathcal{X} = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\right\}$, their dissimilarity is quantified by the distance matrix:

$$\left\{\boldsymbol{D} : d_{ij} = h\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right),\ i, j = 1, \ldots, N\right\}, \tag{3.11}$$

where $h(\cdot, \cdot)$ corresponds to a distance metric.

The goal of maintaining the dissimilarity of the samples in the reduced space is expressed by a suitable loss function (also called *stress function* in the MDS literature). There exist several types of MDS that mainly differ in the stress function and the distance metric $h(\cdot, \cdot)$ that they use.

A classical variant of MDS is the so-called *classical scaling* where $h(\cdot, \cdot)$ corresponds to the Euclidean distance and the loss function reads:

$$J(\mathcal{Z}) = \sum_{i,j=1}^{N} \left(d_{ij}^2 - h\left(\mathbf{z}^{(i)}, \mathbf{z}^{(j)}\right)^2\right). \tag{3.12}$$

Classical scaling specifically looks for a linear mapping $\mathcal{Z} = \mathcal{X}\boldsymbol{M}$ that minimises $J(\mathcal{Z})$. It can be shown that under these assumptions the minimum of $J(\mathcal{Z})$ in Eq. (3.12) is given by the eigendecomposition of the Gram matrix $\boldsymbol{K} = \mathcal{X}\mathcal{X}^\top$ (see *e.g.* Cox and Cox (2000)). The entries of the Gram matrix $\boldsymbol{K}$ can be calculated directly from $\boldsymbol{D}$ as follows (Cox and Cox, 2000):

$$k_{ij} = -\frac{1}{2}\left(d_{ij}^2 - \frac{1}{n}\sum_{l=1}^{N} d_{jl}^2 + \frac{1}{n^2}\sum_{l,k=1}^{N} d_{lk}^2\right). \tag{3.13}$$

The optimal reduced space samples $\mathcal{Z} = \left\{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(N)}\right\}$ are calculated by:

$$\mathcal{Z} = \boldsymbol{\Lambda}^{1/2}\boldsymbol{V}, \tag{3.14}$$

where $\boldsymbol{V}$ corresponds to the collection of the m eigenvectors with the maximal associated eigenvalues contained in $\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_m)$, obtained from the eigen-decomposition of the gram matrix. There is a close connection between classical scaling and PCA that is described in detail in Williams (2002).

In contrast to classical scaling and PCA, different loss functions may lead to a non-linear dimensionality reduction. A commonly used non-linear MDS variant is based on using the *Sammon loss function* that reads:

$$J_S(\mathcal{Z}) = \frac{1}{\sum_{i,j=1}^{N} d_{ij}} \sum_{i,j=1,\, i\neq j}^{N} \frac{\left(d_{ij} - h\left(\mathbf{z}^{(i)}, \mathbf{z}^{(j)}\right)\right)^2}{d_{ij}} \tag{3.15}$$

The Sammon loss function differs from the one in Eq. (3.12) in that it puts more emphasis on retaining distances that were originally small. In this case there is no closed-form solution for calculating the minimiser of Eq. (3.15) and $\mathcal{Z}$ is typically obtained by iterative methods based on gradient descent (Cox and Cox, 2000; Sun et al., 2012).

The main area of application of MDS is the visualisation of multidimensional data, but it can also be used for dimensionality reduction within different contexts such as data compression. Nevertheless, this method has some disadvantages that are discussed next. MDS does not provide an inverse transformation $\mathcal{D}_{\mathbf{Z}} \mapsto \mathcal{D}_{\mathbf{X}}$. In fact, the MDS algorithms that are used

in practice do not even consider the samples $\mathcal{X}$ but only their distance matrix $\boldsymbol{D}$. From a computational perspective, MDS scales poorly with respect to the number of samples in $\mathcal{X}$, having $O(N^3)$ computational complexity and $O(N^2)$ memory requirements. In more complex applications where the samples lie on a curved manifold, MDS using metric distance metrics may show limited compressive performance. This is due to the potentially large discrepancy between the Euclidean and geodesic distance[1] between the samples.

The drawbacks of MDS gave rise to several extensions that have been extensively used in practice. *Isomap* is such an extension that attempts to preserve the pairwise geodesic distances between samples instead of the Euclidean one (Tenenbaum et al., 2000). This allows it to identify non-linear manifolds more robustly. In Isomap, one first constructs a neighbourhood graph in which every sample of $\mathcal{X}$ is connected to its k nearest neighbours. Then the geodesic distance between each pair of samples is approximated by the shortest path between them in the graph using standard shortest path algorithms such as Dijkstra (1959) or Floyd (1962).

A major weakness of Isomap is related to its topological instability. This issue, commonly referred to as *short-circuiting*, has motivated additional methods that focus on the preservation of local properties of each sample in $\mathcal{X}$. In this case, if short-circuiting occurs, only a small subset of the reduced space is affected. Locally linear embedding (Roweis and Saul, 2000) and Laplacian eigenmaps (Belkin and Niyogi, 2002) belong to this class of methods.

It is noteworthy to mention that MDS, and its variants that were discussed so far, do not provide *out-of-sample predictions*, *i.e.* it is not possible to obtain the mapping $\mathbf{z}'$ of a new sample $\mathbf{x}'$ (not contained in $\mathcal{X}$) without recomputing the full reduced representation $\{\mathcal{Z}, \mathbf{z}'\}$ from scratch. However, various approaches for calculating approximate out-of-sample predictions have been proposed in the literature, see *e.g.* Bengio et al. (2004); Strange and Zwiggelaar (2011).

1 The *geodesic* or *curvilinear distance* between two samples lying on a manifold can be loosely defined as the shortest path between them, measured over the manifold.

3.4 KERNEL PRINCIPAL COMPONENT ANALYSIS

Kernel PCA (KPCA) is the reformulation of PCA in a high-dimensional space that is constructed using a kernel function (Schölkopf et al., 1998). A kernel function applied on two elements $\mathbf{x}^{(i)}, \mathbf{x}^{(j)} \in \mathcal{D}_{\mathbf{x}}$ has the following form:

$$\kappa\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right) = \Phi\left(\mathbf{x}^{(i)}\right) \cdot \Phi\left(\mathbf{x}^{(j)}\right), \tag{3.16}$$

where $\Phi(\cdot)$ is a function that performs the mapping $\Phi : \mathcal{D}_{\mathbf{x}} \to \mathcal{H}$ and $\mathcal{H}$ is a Hilbert space, also known as feature space. Based on Eq. (3.16), the so-called *kernel trick* is applied, which refers to the observation that, if the access to $\mathcal{H}$ only takes place through inner products, then there is no need to explicitly define $\Phi(\cdot)$. The result of the inner product can be directly calculated using $\kappa(\cdot, \cdot)$. Kernel PCA is a non-linear extension of PCA where the kernel trick is used to perform PCA in $\mathcal{H}$. The principal components in $\mathcal{H}$ are obtained from the eigen-decomposition of the sample covariance matrix $\mathbf{C}_{\mathcal{H}}$, analogously to the PCA case in Eq. (3.6).

However, in KPCA the eigen-decomposition problem:

$$\mathbf{C}_{\mathcal{H}} \boldsymbol{v}^{(i)} = \lambda_i \boldsymbol{v}^{(i)}, \; i = 1, \ldots, N \tag{3.17}$$

is intractable, since $\mathbf{C}_{\mathcal{H}}$ cannot in general be computed ($\mathcal{H}$ may be infinite-dimensional). This problem is by-passed by observing that each eigenvector belongs to the span of the samples $\Phi\left(\mathbf{x}^{(1)}\right), \ldots, \Phi\left(\mathbf{x}^{(N)}\right)$, therefore scalar coefficients $\alpha_k^{(i)}$ exist, such that each eigenvector $\boldsymbol{v}^{(i)}$ can be expressed as the following linear combination (Schölkopf et al., 1998):

$$\boldsymbol{v}^{(i)} = \sum_{k=1}^{N} \alpha_k^{(i)} \Phi\left(\mathbf{x}^{(k)}\right), \; i = 1, \ldots, N. \tag{3.18}$$

Based on Eq. (3.18) it can be shown that the eigen-decomposition problem in Eq. (3.17) can be cast as:

$$\mathbf{K}\boldsymbol{\alpha}^{(i)} = \lambda^{(i)} \boldsymbol{\alpha}^{(i)}, \; i = 1, \ldots, N, \tag{3.19}$$

where $\mathbf{K}$ is the kernel matrix with elements:

$$K_{ij} = \kappa\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right). \tag{3.20}$$

As for the case of PCA, dimensionality reduction is achieved by only retaining the first m principal axes $\left\{ v^{(i)}, i = 1, \ldots, m \right\}$ onto which $\mathcal{X}$ is projected in order to calculate $\mathcal{Z}$. Schölkopf et al. (1998) showed that $\mathcal{Z}$ can be directly computed based only on the values of the eigenvector expansion coefficients $\alpha_k^{(i)}$ and the kernel matrix $\boldsymbol{K}$. The k-th component of the i-th sample of $\mathcal{Z}$, denoted by $z_k^{(i)}$ is given by:

$$z_k^{(i)} = \Phi\left(\mathbf{x}^{(i)}\right)^{\mathsf{T}} v^{(k)} = \sum_{j=1}^{N} \alpha_k^{(j)} \kappa\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right) . \tag{3.21}$$

The key ingredient of KPCA is arguably the kernel function κ. A list of kernels that are used in this thesis is available in Table 3.1. In the context of KPCA, the isotropic Gaussian kernel (also known as *radial basis function* - RBF) is most often used instead of the anisotropic one by assuming the same parameter value w_k for all components of $\mathbf{x}$. Polynomial kernels provide a connection between KPCA and standard PCA, since KPCA using a polynomial kernel with parameters $w_1 = 1$, $w_2 = 0$ and $w_3 = 1$ is identical to PCA. This is based on the observation that this kernel corresponds to the identity mapping $\Phi(\mathbf{x}) = \mathbf{x}$, hence the covariance matrix $\boldsymbol{C}_{\mathcal{H}}$ in the feature space is identical to the covariance matrix $\boldsymbol{C}$ in the initial space (Eq. (3.6)).

TABLE 3.1: A selection of kernels for KPCA.

Name	**Expression**	**Parameters**
Gaussian	$\kappa(\mathbf{x}, \mathbf{x}'; \mathbf{w}) = \exp\left(-\frac{1}{2}\sum_{k=1}^{M} \frac{1}{w_k^2}\left(x_k - x_k'\right)^2\right)$	$w_k > 0, k = 1, \ldots, M$
Polynomial	$\kappa(\mathbf{x}, \mathbf{x}'; \mathbf{w}) = \left(w_1 \mathbf{x}^{\mathsf{T}} \mathbf{x}' + w_2\right)^{w_3}$	$w_1 > 0$, $w_2 \geq 0$, $w_3 \in \mathbb{N}$
Cosine	$\kappa(\mathbf{x}, \mathbf{x}'; \mathbf{w}) = \frac{\mathbf{x}^{\top} \mathbf{x}'}{\| \mathbf{x} \| \| \mathbf{x}' \|}$	-

From Eq. (3.21) it follows that $\mathcal{Z}$ can be expressed as $\mathcal{Z} = g(\mathcal{X}; \mathbf{w})$ where $\mathbf{w}$ encompasses both the kernel parameters and the reduced space dimension m. In practice, the parameters $\mathbf{w}$ of KPCA are often manually tuned. However, in this thesis two methods to infer their values in an algorithmic fashion are considered, each optimising with respect to a different performance objective.

The *distance preservation* method aims at optimising $\mathbf{w}$ in such a way that the Euclidean distances between the samples are preserved between the original and the feature space (Weinberger et al., 2004). This is expressed by the following objective function:

$$J_{dist}(\mathbf{w}; \mathcal{X}) = \sum_{i,j=1}^{N} \left(d_{ij} - \delta_{ij}\right)^2 , \tag{3.22}$$

where

$$d_{ij} = \left\| \mathbf{x}^{(i)} - \mathbf{x}^{(j)} \right\| , \tag{3.23}$$

and

$$\delta_{ij} = \left\| \Phi(\mathbf{x}^{(i)}, \mathbf{w}) - \Phi(\mathbf{x}^{(j)}, \mathbf{w}) \right\| . \tag{3.24}$$

By expanding the norm expression in Eq. (3.24) it is straightforward to show that:

$$\delta_{ij} = \sqrt{K_{ii} + K_{jj} - 2K_{ij}}, \tag{3.25}$$

hence the value of δ_{ij} is readily available from the kernel matrix $\boldsymbol{K}$.

In contrast, the *reconstruction error*-based method aims at optimising $\mathbf{w}$ so that the so-called pre-image, $\widehat{\mathbf{x}} = g^{-1}(\mathbf{z}, \mathbf{w}')$, of $\mathbf{z} = g(\mathbf{x}, \mathbf{w})$ approximates $\mathbf{x}$ as close as possible (Alam and Fukumizu, 2014). This is expressed by the following objective function:

$$J_{recon}(\mathbf{w}; \mathcal{X}) = \frac{1}{N} \sum_{i=1}^{N} \left\| \mathbf{x}^{(i)} - \widehat{\mathbf{x}}^{(i)} \right\|^2 \tag{3.26}$$

In contrast to PCA, calculating $\widehat{\mathbf{x}}$ is non-trivial, an issue that is known as the *pre-image problem* (see *e.g.* Kwok and Tsang (2003)). Within this thesis, the approximate pre-image is determined using the method proposed Weston et al. (2004) as adopted in practice by Pedregosa et al. (2011). After performing the KPCA transform $\mathcal{D}_{\mathbf{X}} \mapsto \mathcal{D}_{\mathbf{Z}}$, the (non-unique) pre-image of a new point $\mathbf{z}$ is computed by kernel-ridge regression using a separate kernel function κ_{pre}:

$$\widehat{\mathbf{x}} = \boldsymbol{\beta}^{\mathsf{T}} \boldsymbol{l}(\mathbf{z}), \tag{3.27}$$

where:

$$\boldsymbol{\ell}(\mathbf{z}) = \left\{ \kappa_{pre}(\mathbf{z}, \mathbf{z}^{(j)}),\ j = 1, \ldots, N \right\}, \tag{3.28}$$

and $\boldsymbol{\beta}$ are the kernel-ridge regression coefficients. They are calculated as follows:

$$\boldsymbol{\beta} = (\boldsymbol{L} + r\boldsymbol{I}_N)^{-1} \mathcal{X} \qquad L_{ij} = \left\{ \kappa_{pre} \left(\mathbf{z}^{(i)}, \mathbf{z}^{(j)} \right),\ i, j = 1, \ldots, N \right\}, \quad (3.29)$$

where r is a regularisation parameter and $\boldsymbol{I}_N$ is the N-dimensional identity matrix. Following the approach in Pedregosa et al. (2011), we use for simplicity the same kernel for the pre-image problem as for KPCA, *i.e.* $\kappa_{pre}(\cdot, \cdot)$ is chosen equal to $\kappa(\cdot, \cdot)$.

Note that, in the unsupervised learning literature, the target dimension m of the reduced space is not part of $\mathbf{w}$, *i.e.* only the kernel parameters are considered when minimising the objective function in Eq. (3.22) or Eq. (3.26).

Kernel PCA has been applied successfully to problems such as face recognition (Ming-Hsuan Yang and Kriegman, 2000), speech recognition (Lima et al., 2003) and novelty detection (Hoffmann, 2007). A major weakness of the method is that the kernel matrix size is proportional to N^2 and it needs to be stored in memory. Additionally the computational complexity of KPCA is $O(N^3)$ so the method in its standard form does not scale well with increasing sample size. Parameter tuning may also become an issue depending on the kernel function that is used. Notice that the kernel selection may be included in the optimisation step, *i.e.* the kernel type can be included in the parameter vector $\mathbf{w}$.

KPCA can be characterized as a modular dimensionality reduction with different behaviour depending on the kernel function that it uses. Ghodsi (2006) shows that MDS with Euclidean distance metric, Isomap and other dimensionality reduction methods can be expressed as KPCA with a particular kernel.

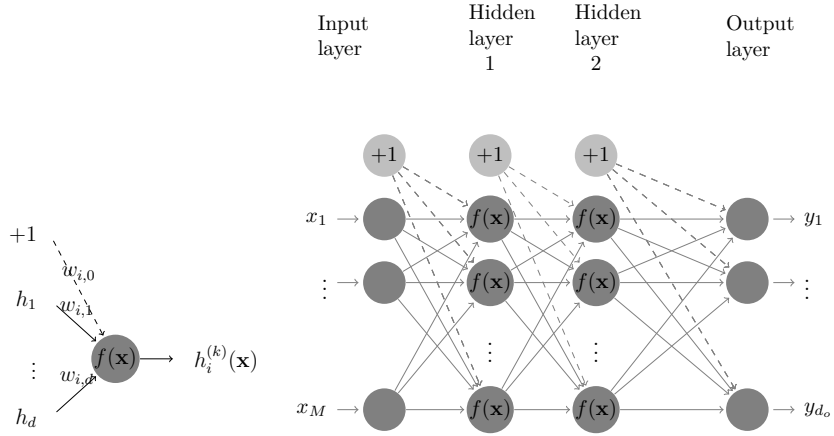
Several extensions of KPCA have been proposed for reducing its memory requirements and computational complexity. For reducing the memory requirements, Tipping (2001) suggests to apply KPCA by sparse projections based on the approximation of the covariance matrix $\boldsymbol{C}_{\mathcal{H}}$ by a subset of outer products of feature vectors. More recently, various techniques have been proposed for performing KPCA in an iterative fashion so that the method may be applied on a data stream where each sample of $\mathcal{X}$ is

processed only once by the algorithm (Kuzmin and Warmuth, 2007; Günter et al., 2007).

3.5 NEURAL NETWORKS

Artificial neural networks (ANNs) stand for a group of methods the functional form of which is represented by a graph that is vaguely inspired by biological neural networks. They have seen rapid evolution from the first introduction of their concept in McCulloch and Pitts (1943) to their recent popular variants commonly used in *deep learning* (Goodfellow et al., 2016). A general overview of ANNs is given next, before discussing in more detail particular ANNs that are used for dimensionality reduction.

The main building block of any ANN is a node, also known as a *neuron*. Edges, called *weights*, connect neurons with each other. Different types of ANNs involve different types of neurons and weights that result in a different behaviours.



(a) An ANN node (neuron). (b) A feed-forward ANN with two hidden layers.

FIGURE 3.1: Graph representation of feed-forward ANNs.

A specific family of ANNs is mainly considered in this work, called *feed-forward* neural networks or *multilayer perceptrons*. The main building block of such ANN is the neuron depicted in Figure 3.1a. The neurons are organised in groups, called *layers*. An example feed-forward ANN is shown

in Figure 3.1b. Each neuron (node) i, that belongs to the layer k of the network, performs the operation:

$$h_i^{(k)} = f\left(\sum_{j=0}^{d_{k-1}+1} w_{i,j}^{(k)}\, h_j^{(k-1)}\right), \tag{3.30}$$

where $h_i^{(k)}$ denotes the output value of the i-th neuron that belongs to the k-th layer, d_{k-1} corresponds to the number of neurons in the $(k-1)$-th layer, $w_{i,j}^{(k)} \in \mathbb{R}$ is the weight of the connection between the node i of the k-th layer and the node j of the $(k-1)$-th layer. The function $f(\cdot)$ in Eq. (3.30) is called the *activation function*. It is a scalar function $f : \mathbb{R} \to \mathbb{R}$ that often belongs to the sigmoid family. Common choices of activation functions in this class of ANNs are:

$$f(x) = \frac{1}{1+e^{-x}}\,, \tag{3.31}$$

or

$$f(x) = \tanh(x)\,. \tag{3.32}$$

In more recent applications, non-sigmoidal activation functions are also used, such as the so-called *rectified linear unit* (ReLU) (Goodfellow et al., 2016):

$$f(x) = \max(0, x)\ . \tag{3.33}$$

Consider an ANN such as the one in Figure 3.1b. A sample $\mathbf{x}$ is processed first by the nodes in the input layer, which is the leftmost layer. The nodes of that layer have no activation function, and only a single input corresponding to a component of $\mathbf{x}$. Based on Eq. (3.30), it is straightforward to see that their output is the identity function, *i.e.* $\left\{h_i^{(0)} = x_i\,,\ i = 1, \ldots, M\right\}$. Each node of the input layer is connected to all the nodes of the next layer, called a *hidden layer*. An ANN can contain an arbitrary number of hidden layers. The nodes of each layer are fully connected to all the nodes of the previous one. Notice that in each layer there exists a special node, called the *bias* unit and denoted by $+1$, that constantly outputs the value 1. It is useful

in practice because it enables affine transformations to the output values of each layer leading to improved performance (Hagan et al., 1996). The final layer of the ANN is called the *output layer*. Its nodes differ from the ones in the hidden layers only in terms of the activation function that is used. In this example there is no activation function to the output layer neurons.

The l-th output of the ANN in Figure 3.1b is calculated by *forward propagation* that reads:

$$y_l = \sum_{k=0}^{d_2} w_{l,k}^{(3)} \, f\left(\sum_{j=0}^{d_1} w_{k,j}^{(2)} \, f\left(\sum_{i=0}^{M} w_{j,i}^{(1)} \, x_i \right) \right) , \, l = 1, \ldots , d_o \, , \tag{3.34}$$

where d_o corresponds to the number of neurons in the output layer.

Clearly, such ANNs produce a non-linear input-output relationship. In fact, it has been shown that feed-forward ANNs with a single hidden layer can approximate any bounded, continuous functional input-output relationship (Hornik et al., 1989; Leshno et al., 1993). However, the required hidden layer nodes might tend to infinity. Various studies have concluded that by introducing additional hidden layers the learning capacity of a feed-forward ANN is improved more efficiently, (*i.e.* requires a smaller total number of neurons) compared to the case of having a single hidden layer (Montufar et al., 2014; Goodfellow et al., 2016).

In this work, the focus is given on specific feed-forward ANN architectures that are used in the context of dimensionality reduction, called *autoencoders*. An autoencoder with a single hidden layer is shown in Figure 3.2. Multiple hidden layers may exist, that lead to the so-called *stacked* autoencoders. The goal of an autoencoder is to learn the identity function, *i.e.* reproduce the values of the input $\mathbf{x}$ in the output $\widehat{\mathbf{x}}$. The number of nodes that correspond to the *encoding* step are decreasing in each successive layer enforcing an *information bottleneck*. The layers that follow the middle hidden layer form the *decoding* step that perform the opposite operation: transforming back the reduced representation of the input to the input itself. The middle layer of the autoencoder, *i.e.* the bottleneck of the identity function that is learned, forms the representation of the reduced space $\mathbf{Z}$.

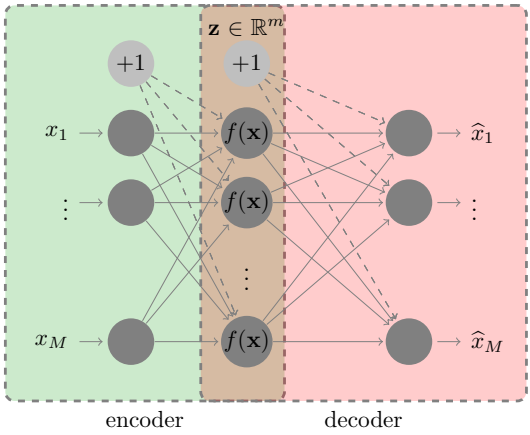


FIGURE 3.2: An autoencoder with a single hidden layer.

The reduced space representation is of the form $\mathbf{z} = g(\mathbf{x}; \mathbf{W}_{enc})$, where $\mathbf{W}_{enc}$ corresponds to the collection of the weights in each layer up to the middle one. For clarity, only the weights are considered as unknown parameters, but in general the architecture of the autoencoder could also be included (*i.e.* the number of layers and the number of nodes per layer). The unknown parameters $\mathbf{W}$ are calibrated by minimising an objective function related to the reconstruction error such as the following (Vapnik, 1998):

$$J_{recon}(\mathbf{W}; \mathcal{X}) = \frac{1}{2} \sum_{i=1}^{N} \left\| \mathbf{x}^{(i)} - \widehat{\mathbf{x}}(\mathbf{W}; \mathbf{x}^{(i)}) \right\|^2 , \tag{3.35}$$

where $\mathbf{W} = \{\mathbf{W}_{enc}, \mathbf{W}_{dec}\}$ is the collection of all the weights of the ANN (both the encoding and decoding parts) and $\widehat{\mathbf{x}}(\mathbf{W}; \mathbf{x}^{(i)})$ is the i-th output of the network calculated by forward propagation, similarly to Eq. (3.34). In ANN terminology, the minimisation of $J_{recon}(\mathbf{W}; \mathcal{X})$ is called *training* of the ANN.

Although ANNs can be considered universal function approximators there is a caveat. It may be theoretically guaranteed that a solution of Eq. (3.35) exists but it can be challenging or even impossible to calculate it in practice. Due to the non-linearity of the activation function $f(\cdot)$, the minimisation of Eq. (3.35) is a non-convex optimization problem characterized by local

minimas and flat regions. Various implementations of autoencoders include different regularisation schemes aiming to improve the performance of the optimizer.

The seminal paper by Hinton and Salakhutdinov (2006) that gave rise to the deep learning concept suggests a pre-training procedure. This involves an approximate, low-cost training of a binary representation of the ANN that is called *restricted Boltzmann machine*. Sparse autoencoders refer to variants that include a penalty term in Eq. (3.35) to enforce sparsity in the output values of the hidden layer nodes (Boureau et al., 2008; Glorot et al., 2011). Sparsity in the reduced space representation can be beneficial for various reasons. For example, Hinton et al. (2006) showed that pre-training an ANN to learn a sparsified representation resulted in a significantly more efficient subsequent training of the same ANN for performing a classification task. Moreover, in the context of pattern recognition in image processing, Goodfellow et al. (2009) found that sparse autoencoders gave rise to more invariant representations (compared to non-sparse ones), in the sense that a subset of the representation elements were more insensitive to input transformations such as translation or rotation of the camera.

Denoising autoencoders introduce noise corruption on the input samples instead of adding a penalty term (Vincent et al., 2008; Chen et al., 2014). Contractive autoencoders introduce a penalty term proportional to $\| \nabla_{\mathbf{x}} h_i^{(k)} \|^2$, forcing the model to learn a function with minimal sensitivity to small variations in the value of $\mathbf{x}$ (Rifai et al., 2011).

In this chapter, the autoencoder implementation provided by the MATLAB software is used (MathWorks Inc., 2017). It considers an extended version of Eq. (3.35) that reads:

$$J_{recon}(\mathbf{W};\mathcal{X}) = \frac{1}{2}\sum_{i=1}^{N}\left\|\mathbf{x}^{(i)} - \hat{\mathbf{x}}(\mathbf{W};\mathbf{x}^{(i)})\right\|^2 + \lambda_s\Omega_s(\mathbf{W}) + \lambda_n \|\mathbf{W}\|^2, \quad (3.36)$$

where Ω_s is a sparsity regularisation term, $\|\mathbf{W}\|^2$ is the L2-norm penalisation of the autoencoder's weights and λ_s, λ_n serve as the respective weights of each term. The L2 (or L1) -norm regularisation is a common practice in regression problems, not just autoencoders, to avoid over-fitting (see *e.g.*

Vapnik (1998)). The term Ω_s enforces a constraint on the activation sparsity of the hidden layer nodes. Activation sparsity refers to how often each node activates (has non-zero value) over the training samples $\mathcal{X}$. The sparsity penalisation that is used in the MATLAB autoencoder implementation is based on the *Kullback-Leibler divergence* and reads (Olshausen and Field, 1997):

$$\Omega_s = \sum_{i=1}^{m} KL(\rho||\hat{\rho}_i) \stackrel{\text{def}}{=} \sum_{i=1}^{m} \rho \log\left(\frac{\rho}{\hat{\rho}_i}\right) + (1-\rho)\log\left(\frac{1-\rho}{1-\hat{\rho}_i}\right), \qquad (3.37)$$

where ρ is a parameter that determines the desired average activation and, in general, $\hat{\rho}_i^{(k)}$ corresponds to the average activation of the i-th neuron in the k-th hidden layer:

$$\hat{\rho}_i^{(k)} = \frac{1}{N}\sum_{j=1}^{N} h_i^{(k)}(\mathbf{x}^{(j)}). \qquad (3.38)$$

In Eq. (3.37) only the middle hidden layer of the autoencoder is considered, hence the layer index in Eq. (3.38) is dropped for notational clarity. Note that this type of sparsity regularisation assumes a bounded activation function in $[0,1]$. This is true for the sigmoid function in Eq. (3.31) and straightforward to extend for Eq. (3.32) by rescaling its output from $[-1,1]$ to $[0,1]$ but not for ReLU activation (Eq. (3.33)) because it is unbounded.

Given the architecture of the autoencoder and the loss function, its weights are calculated by minimising that loss function. Well-established algorithms such as stochastic gradient descent (SGD) are used in practice, where the network weights are updated based on the approximate gradient of the objective function after processing a batch of the samples. For an overview of modern optimisation algorithms that are used in the context of ANNs see *e.g.* Goodfellow et al. (2016).

Autoencoders have been successfully used for dimensionality reduction (Hinton and Salakhutdinov, 2006), image de-noising (Xie et al., 2012) and as a pre-training tool of ANNs for classification problems (Rifai et al., 2011). In addition, a popular variant in the context of image processing is the

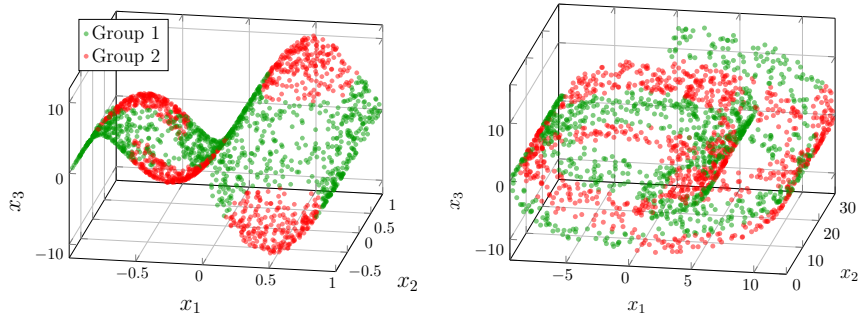
so-called *convolutional autoencoders* (Theis et al., 2017). Their architecture differs from feed-forward ANNs and is based on convolution operations in each layer (see *e.g.* Goodfellow et al. (2016)). The main drawback of autoencoders is related to the tuning of numerous parameters, including the structure of the network (number of layers, number of nodes per layer) and the parameters of the optimisation algorithm. Although some heuristic guidelines exist, such as those given in Hinton (2012), it is often non-trivial and time consuming to tune them.

An appealing feature of this method is that its computational complexity scales linearly with respect to the sample size N and the size of the weight vector $\mathbf{W}$. In addition, its memory requirements are low because only the weight matrix needs to be stored. Finally, in contrast to the previously presented non-linear dimensionality reduction methods, a trained autoencoder can easily provide the (approximate) inverse transformation $g^{-1} : \mathcal{D}_{\mathbf{Z}} \to \mathcal{D}_{\mathbf{X}}$ by forward propagation from the middle hidden layer to the output layer.

3.6 COMPARISON OF THE METHODS ON BENCHMARK DATASETS

To showcase the dimensionality reduction methods that have been presented so far, two benchmark datasets are considered, namely the *twin peaks* (Figure 3.3a) and the *Swiss roll* dataset (Figure 3.3b). They are provided for benchmarking purposes by the "MATLAB Toolbox for Dimensionality Reduction" by Van Der Maaten et al. (2009). In both cases the goal of the comparison is to assess the information loss due to dimensionality reduction. One measure for such assessment is the reconstruction error in Eq. (3.5). However, due to the lack of inverse transform $g^{-1} : \mathcal{D}_{\mathbf{Z}} \to \mathcal{D}_{\mathbf{X}}$ for some of the methods, a qualitative assessment also takes place by inspecting the reduced space that is obtained by each of them.

In both datasets, $\mathbf{X}$ lies on a 2-dimensional manifold that is embedded in the 3-dimensional space. Hence, each of the dimensionality reduction approaches that were presented in the previous sections is used to compress the input from 3 to 2 dimensions. Further comparisons are facilitated by assigning to each sample in both datasets a binary label. This allows one to observe how the neighbourhood of each sample is affected by the dimensionality reduction step.



(a) Twin peaks.

(b) Swiss roll.

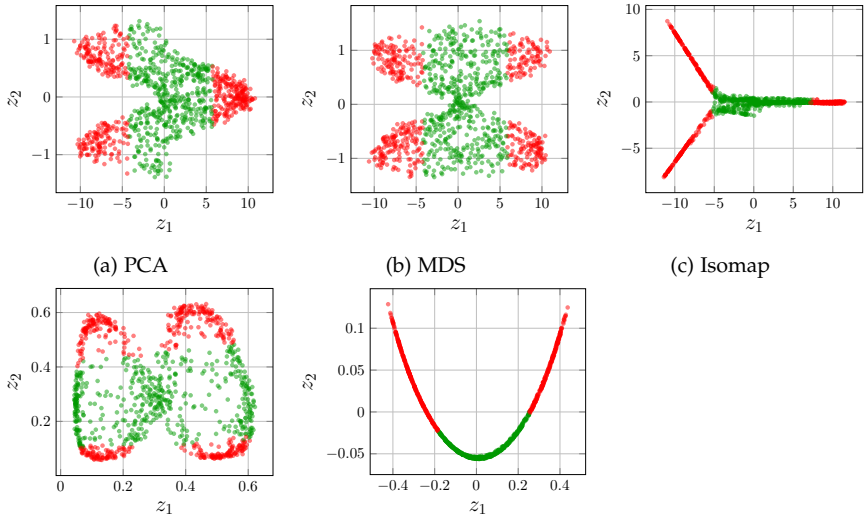
FIGURE 3.3: The benchmark datasets.

TABLE 3.2: Selected parameters for each dimensionality reduction method used to compress the twin peaks and Swiss roll dataset.

Method	**Parameters**		
	Name	Value	
		Twin peaks	Swiss roll
PCA	-	-	-
MDS	Cost function	Shammon (Eq. (3.15))	
Isomap	Num. of neighbours	12	12
Stacked autoencoder	Architecture (encoder)	$(30, 15, 5, 2)$[1]	$(30, 20, 10, 2)$
	Note: The decoder architecture is symmetric to the encoder's		
	Regularisation	As in Eq. (3.36)	
	λ_s (Eq. (3.36))	0.1	0.1
	ρ (Eq. (3.36))	0.3	0.3
	λ_n (Eq. (3.36))	10^{-4}	10^{-4}
	Activation function	Sigmoid (Eq. (3.31)) in encoder neurons and linear ($f(x) = x$) in decoder neurons	
Kernel PCA	Kernel type	RBF	Cosine
	Kernel parameters	$\sigma = 0.001$	-
	r (Eq. (3.29))	1	1

1 A compact way to describe the architecture of a feed-forward ANN is by using *e.g.* the notation (a, b, c) when the ANN consists of three layers each having a number of nodes that is equal to

Each dimensionality reduction technique is applied on $N = 2,000$ samples for each dataset, using the parameters reported in Table 3.2.



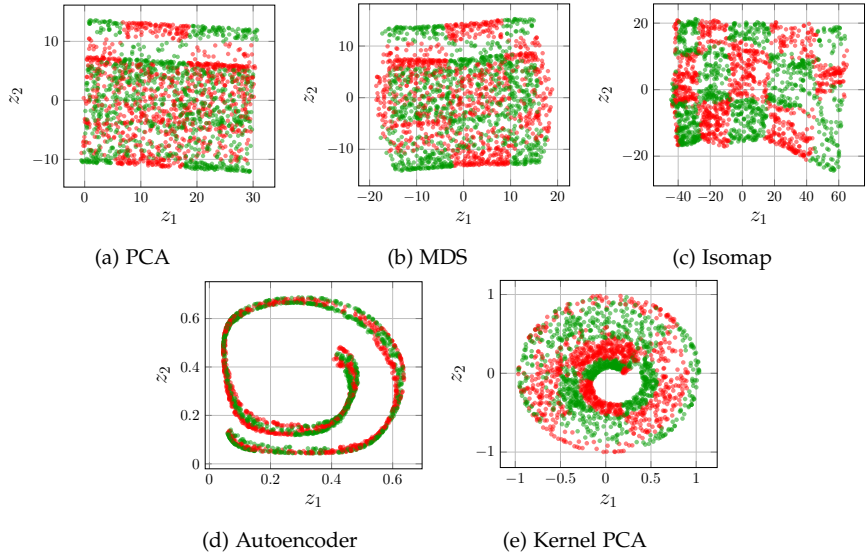
(a) PCA (b) MDS (c) Isomap

(d) Autoencoder (e) Kernel PCA

FIGURE 3.4: Dimensionality reduction results on the twin peaks dataset.

The compressed spaces that were obtained by each method for the twin peaks dataset are shown in Figure 3.4. At a first glance, the reduced space in each case is remarkably different. The four highlighted peaks in the original space are clearly maintained by MDS in Figure 3.4b and the autoencoder in Figure 3.4d. In both cases a clear separation is shown between the samples that belong to one of the peaks and the rest. Notice that such representations can be useful *e.g.* for building a classifier between the two regions. PCA in Figure 3.4a "unfolds" three out of the four peaks with two overlapping in the half-plane $z_1 > 5$. Finally, Isomap in Figure 3.4c and Kernel PCA in Figure 3.4e separate the peak regions from the rest but show significant overlaps. Depending how the compressed space is used later on, this may or may not be a disadvantage. For example, a classifier between the two regions could perform well on these spaces but in the context of visualisation there is plenty of information lost regarding the relative position of each sample with respect to its nearest neighbours.

a, *b* and *c*, respectively. The input layer can be excluded in this notation because the number of nodes is clearly M (the dimensionality of the physical space).



(a) PCA (b) MDS (c) Isomap

(d) Autoencoder (e) Kernel PCA

FIGURE 3.5: Dimensionality reduction results on the Swiss roll dataset.

The same comparison is conducted next on the Swiss roll dataset (Figure 3.5), that corresponds to an arguably more complex manifold. One may quickly observe that there are significant differences in the quality of the reduced space that each method provides compared to the previous case. PCA (Figure 3.5a) and MDS (Figure 3.5b) show significant overlap between non-neighbouring samples in the original space. Instead, Isomap in Figure 3.5c provides the clearest separation of each region. However, on the left boundary it can be observed that there exist some miss-placed samples. This can be attributed to the main drawback of Isomap that was discussed in Section 3.3, that is related to erroneous connections in the graph.

There exists some similarity between the compressed space obtained by the autoencoder in Figure 3.5d and kernel PCA in Figure 3.5e in the sense that both retained the spiral shape of the Swiss roll which, however, is irrelevant to the true 2D manifold. Kernel PCA provides clear separation between the different regions near the origin, but some spurious neighbourhoods are introduced towards the boundaries of the manifold. On the contrary,

the autoencoder produced a space with significant overlaps of different regions.

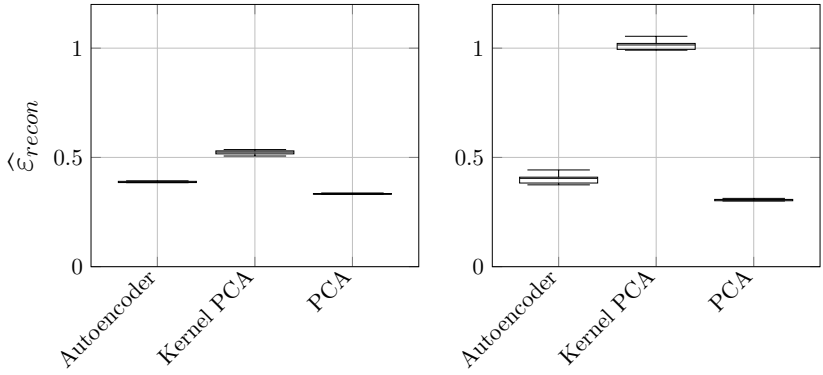
The qualitative comparison between the methods indicated that (i) there is no "Jack of all trades" and each one may produce significantly different results depending on the problem, and, (ii) depending on the context (*e.g.* visualisation, classification) the preferred method may vary.

It is of interest to further compare each dimensionality reduction approach based on their ability to reconstruct the original samples from the compressed ones. In this comparison, MDS and its variant, Isomap, are omitted because they do not provide an inverse transformation. The other methods are compared as follows: for each dataset (twin peaks, Swiss roll) a set of $2,000$ samples is randomly generated and used for calibration. Then a validation set $\mathcal{X}_v = \left\{\mathbf{x}_v^{(1)}, \ldots, \mathbf{x}_v^{(N_v)}\right\}$ with $N_v = 10,000$ samples is randomly generated to evaluate the normalised empirical reconstruction error that reads:

$$\widehat{\varepsilon}_{recon} = \frac{1}{M}\sum_{i=1}^{M}\frac{\sum_{j=1}^{N_v}\left(x_{v,i}^{(j)} - \widehat{x}_j^{(i)}\right)^2}{\sum_{j=1}^{N_v}\left(x_{v,i}^{(j)} - \widehat{\mu}_{v,i}\right)^2}, \tag{3.39}$$

where $x_{v,i}^{(j)}$ denotes the $i-th$ component of the validation set sample $\mathbf{x}_v^{(j)}$, $\widehat{x}_j^{(i)}$ the corresponding component of the reconstructed sample and $\widehat{\mu}_{v,i}$ the sample mean of the i-th component of $\mathbf{x}$ evaluated on the validation set. The training and validation samples are generated at random 10 times in order to observe how $\widehat{\varepsilon}_{recon}$ varies using the same configuration for each dimensionality reduction method as reported in Table 3.2.

In Figure 3.6 a comparison between the reconstruction error that each method achieves is made using box plots. Each box provides summary statistics of the reconstruction error that was achieved by each method over the 10 repetitions. The central mark indicates the median and the bottom and top edges indicate the 25^{th} and 75^{th} percentiles, respectively. The whiskers correspond to 1.5 times the inter-quartile range centered on the median. The box plots in Figure 3.6a and Figure 3.6b correspond to the results that were obtained on the twin peaks and the Swiss roll dataset



(a) Twin peaks (b) Swiss roll

FIGURE 3.6: Reconstruction error comparison.

respectively. At a first glance, PCA consistently outperformed the others in both datasets both in terms of the reconstruction error achieved, as well as the robustness of the results over different repetitions. Autoencoders follow with slightly reduced but comparable performance to PCA and somewhat less robust on the Swiss roll dataset. However, due to the numerous parameters involved in the tuning process of an autoencoder, it is impossible to guarantee that this the best performance that they can achieve. Finally, kernel PCA showed the worst performance for both datasets and by a significant margin in the Swiss roll case. This can be attributed to the pre-image problem and the approximate nature of the inverse transformation that was discussed in Section 3.4.

3.7 CONCLUSION

High-dimensional data often occur in practice and need to be compressed before further processing. To deal with such data, the concept of dimensionality reduction was introduced in this chapter. There exists a broad spectrum of relevant applications, such as data visualisation, general purpose data compression and image de-noising. Various parametric dimensionality reduction techniques have been presented each with its own strengths and weaknesses. Regardless of the specificity of each method, it was shown that they can be expressed as a transform $g : \mathcal{D}_{\mathbf{X}} \to \mathcal{D}_{\mathbf{Z}}$ of the form $\mathbf{Z} = g(\mathbf{X}; \mathbf{w})$ where the parameters $\mathbf{w}$ of the transform are inferred from the available

data $\mathcal{X}$ within a data-driven context. Selecting a suitable method depends on a number of factors including the complexity of the problem, the number of available samples and most importantly the end-goal (visualisation, compression, classification, etc.).

In the next chapters, dimensionality reduction will be optimally combined with surrogate modelling. This new type of application differs from the various usage scenarios presented above and will be described in all details in Chapter 5.

4

SURROGATE MODELLING

4.1 INTRODUCTION

In the context of uncertainty quantification (UQ), the physical or computational model of a system can be seen as a black-box that performs the mapping:

$$\boldsymbol{Y} = \mathcal{M}(\boldsymbol{X}), \tag{4.1}$$

where $\boldsymbol{X}$ is a random vector that parametrises the uncertainty of the input parameters (*e.g.* through a joint probability density function) and $\boldsymbol{Y}$ is the corresponding random vector of model responses. One of the main applications of UQ is to propagate the uncertainties from $\boldsymbol{X}$ to $\boldsymbol{Y}$ through the model $\mathcal{M}$. Direct methods based on Monte-Carlo simulation can easily require that the computational model is run thousands to millions of times for different realisations $\mathbf{x}$ of the input random vector $\boldsymbol{X}$. However, most models that are used in applied sciences and engineering (*e.g.* high-resolution finite element models) can have high computational costs per model run. As a consequence, they cannot be used directly. To alleviate the associated computational burden, surrogate models have become a staple tool in all types of uncertainty quantification applications.

A surrogate model $\widehat{\mathcal{M}}$ is a computationally inexpensive approximation of the true model of the form:

$$\mathcal{M}(\boldsymbol{X}) = \widehat{\mathcal{M}}(\boldsymbol{X}; \boldsymbol{\theta}) + \epsilon, \tag{4.2}$$

where $\boldsymbol{\theta}$ is a set of parameters that characterise the surrogate model and ϵ refers to an error term. The parameters $\boldsymbol{\theta}$ are inferred (typically through some form of optimisation process) from a limited set of runs of the original model $\mathcal{X} = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\right\}$, called the *experimental design*. As an example, $\boldsymbol{\theta}$ denotes the set of coefficients to compute in the case of a truncated polynomial chaos expansion, or the set of parameters of both the trend and

the covariance kernel in case of Gaussian process modelling. Throughout the rest of the chapter, the output of the model $\mathcal{M}$ is considered scalar, *i.e.* $y = \mathcal{M}(\mathbf{x}) \in \mathbb{R}$.

Popular surrogate modelling techniques include Gaussian process modelling (also known as *Kriging*) and regression (Sacks et al., 1989; Rasmussen and Williams, 2006), polynomial chaos expansions (Ghanem and Spanos, 1991; Xiu and Karniadakis, 2002; Xiu, 2010), low-rank tensor approximations (Chevreuil et al., 2015; Konakli and Sudret, 2016b), and support vector regression (Vapnik, 1995).

In this thesis we focus on two techniques, namely Kriging and polynomial chaos expansions that are presented next, in Section 4.2 and Section 4.3, respectively. This is followed by a discussion in Section 4.4 on the error measures that are commonly used to assess the performance of a surrogate, regardless of its type. Finally, we showcase the behaviour of Kriging and polynomial chaos expansions on a benchmark application in Section 4.5.

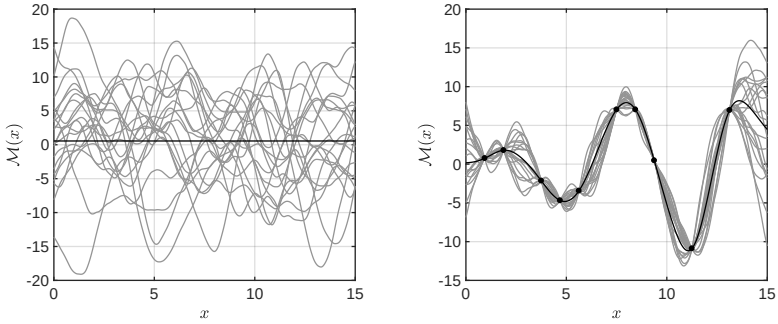
4.2 KRIGING

Kriging is a meta-modelling technique which assumes that the true model response is a realisation of a Gaussian process, described by the following equation (Santner et al., 2003):

$$\mathcal{M}^K(\mathbf{x}) = \boldsymbol{\beta}^\top \boldsymbol{f}(\boldsymbol{x}) + \sigma^2 Z(\mathbf{x}, \omega) \tag{4.3}$$

where $\boldsymbol{\beta}^\top \boldsymbol{f}(\boldsymbol{x})$ is the mean value of the Gaussian process, also called *trend*, σ^2 is the Gaussian process variance and $Z(\mathbf{x}, \omega)$ is a zero-mean, unit-variance Gaussian process. This process is fully characterised by the auto-correlation function between two sample points $R(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta})$. The hyper-parameters $\boldsymbol{\theta}$ associated with the correlation function $R(\cdot; \boldsymbol{\theta})$ are typically unknown and need to be estimated from the available observations.

Having specified the trend and the correlation function parameters it is possible to obtain an arbitrary number of realisations of the so-called *prior* Gaussian process (see Figure 4.1a). In the context of metamodelling the goal is to calculate a prediction $\mathcal{M}^K(x)$ for a new point $\mathbf{x}$, given an experimental



(a) Prior (b) Posterior

FIGURE 4.1: Realisations of a prior and posterior Gaussian process. The Gaussian process mean in each case is denoted by a black line.

design $\mathcal{X} = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\right\}$ of size N and the corresponding (noise-free) model responses $\boldsymbol{y} = \{y^{(1)} = \mathcal{M}(\mathbf{x}^{(1)}), \ldots, y^{(N)} = \mathcal{M}(\mathbf{x}^{(N)})\}^\top$. A Kriging metamodel (a.k.a. Kriging predictor) provides such predictions based on the properties of the so-called *posterior* Gaussian process conditioned on the available data (see Figure 4.1b).

The Kriging prediction on $\mathbf{x}$ corresponds to a random variate $\widehat{Y}(\mathbf{x}) \sim \mathcal{N}\left(\mu_{\widehat{Y}}(\boldsymbol{x}), \sigma_{\widehat{Y}}(\boldsymbol{x})\right)$, therefore the approximation of the computational model that is obtained is essentially an infinite family of such models. Each of those models is a realisation (or sample) of the posterior Gaussian process. In practice, the mean response is used (see Eq. (4.8)) as the Kriging surrogate, while its variance (see Eq. (4.9)) is often interpreted as a measure of the prediction uncertainty. The equations for calculating the mean and variance of a universal Kriging predictor are given next.

The Gaussian assumption states that the vector formed by the true model responses, $\boldsymbol{y}$, and the prediction, $\widehat{Y}(\mathbf{x})$, has a joint Gaussian distribution defined by:

$$\begin{Bmatrix} \widehat{Y}(\mathbf{x}) \\ \boldsymbol{y} \end{Bmatrix} \sim \mathcal{N}_{N+1}\left(\begin{Bmatrix} \boldsymbol{f}^\top(\mathbf{x})\boldsymbol{\beta} \\ \boldsymbol{F}\boldsymbol{\beta} \end{Bmatrix}, \sigma^2 \begin{Bmatrix} 1 & \boldsymbol{r}^\top(\mathbf{x}) \\ \boldsymbol{r}(\mathbf{x}) & \boldsymbol{R} \end{Bmatrix}\right) \tag{4.4}$$

where $\boldsymbol{F}$ is the information matrix of generic terms:

$$F_{ij} = f_j(\boldsymbol{x}^{(i)}) \ , \ i = 1, \ldots, N, \ j = 1, \ldots, P, \tag{4.5}$$

$\boldsymbol{r}(\mathbf{x})$ is the vector of cross-correlations between the prediction point $\mathbf{x}$ and each one of the observations whose terms read:

$$r_i(\mathbf{x}) = R(\mathbf{x}, \mathbf{x}^{(i)}; \boldsymbol{\theta}), \ i = 1, \ldots, N. \tag{4.6}$$

$\boldsymbol{R}$ is the correlation matrix given by:

$$R_{ij} = R(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}; \boldsymbol{\theta}), \ i, j = 1, \ldots, N. \tag{4.7}$$

The mean and variance of the Gaussian random variate $\widehat{Y}(\mathbf{x})$ (a.k.a. mean and variance of the Kriging predictor) can be calculated based on the best linear unbiased predictor properties (Santner et al., 2003):

$$\mu_{\widehat{Y}}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x})^\top \boldsymbol{\beta} + \boldsymbol{r}(\boldsymbol{x})^\top \boldsymbol{R}^{-1} \left(\boldsymbol{y} - \boldsymbol{F}\boldsymbol{\beta}\right) \ , \tag{4.8}$$

$$\sigma^2_{\widehat{Y}}(\boldsymbol{x}) = \sigma^2 \left(1 - \boldsymbol{r}^\top(\boldsymbol{x}) \boldsymbol{R}^{-1} \boldsymbol{r}(\boldsymbol{x}) + \boldsymbol{u}^\top(\boldsymbol{x}) (\boldsymbol{F}^\top \boldsymbol{R}^{-1} \boldsymbol{F})^{-1} \boldsymbol{u}(\boldsymbol{x})\right) \ , \tag{4.9}$$

where:

$$\boldsymbol{\beta} = \left(\boldsymbol{F}^\top \boldsymbol{R}^{-1} \boldsymbol{F}\right)^{-1} \boldsymbol{F}^\top \boldsymbol{R}^{-1} \boldsymbol{y} \, , \tag{4.10}$$

is the generalised least-squares estimate of the underlying regression problem and

$$\boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{F}^\top \boldsymbol{R}^{-1} \boldsymbol{r}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x}). \tag{4.11}$$

It is important to note that the Kriging model interpolates the data, *i.e.*:

$$\mu_{\widehat{Y}}(\mathbf{x}) = \mathcal{M}(\mathbf{x}), \quad \sigma^2_{\widehat{Y}}(\mathbf{x}) = 0, \quad \forall \mathbf{x} \in \mathcal{X} \, . \tag{4.12}$$

Once $\mu_{\widehat{Y}}(\boldsymbol{x})$ and $\sigma^2_{\widehat{Y}}(\boldsymbol{x})$ are available, confidence bounds on predictions can be derived by observing that:

$$\mathcal{P}\left[\widehat{Y}(\boldsymbol{x}) \leq t\right] = \Phi\left(\frac{t - \mu_{\widehat{Y}}(\boldsymbol{x})}{\sigma_{\widehat{Y}}(\boldsymbol{x})}\right), \tag{4.13}$$

where $\Phi(\cdot)$ denotes the Gaussian cumulative distribution function. Based on Eq. (4.13) the confidence intervals on the predictor can be calculated by:

$$\widehat{Y}(\boldsymbol{x}) \in \left[\mu_{\widehat{Y}}(\boldsymbol{x}) - \Phi^{-1}\left(1-\frac{\alpha}{2}\right)\sigma_{\widehat{Y}}(\boldsymbol{x}) \;,\; \mu_{\widehat{Y}}(\boldsymbol{x}) + \Phi^{-1}\left(1-\frac{\alpha}{2}\right)\sigma_{\widehat{Y}}(\boldsymbol{x})\right] , \tag{4.14}$$

and can be interpreted as the interval within which the Kriging prediction falls with probability $1-\alpha$.

The equations that were derived for the best linear unbiased Kriging predictor assumed that the covariance function $\sigma^2 R(\cdot;\boldsymbol{\theta})$ is known. In practice however, the family and other properties of the correlation function need to be selected a priori. The hyperparameters $\boldsymbol{\theta}$, the regression coefficients $\boldsymbol{\beta}$ and the variance σ^2 need to be estimated based on the available experimental design. This involves solving an optimisation problem that is further discussed in Section 4.2.3. The resulting best linear unbiased predictors are called *empirical* in Santner et al. (2003) because they typically result from empirical choice of various Kriging parameters that are further discussed in Sections 4.2.1 - 4.2.3.

4.2.1 *Trend*

The trend refers to the mean of the Gaussian process, *i.e.* the $\boldsymbol{\beta}^\top \boldsymbol{f}(\boldsymbol{x})$ term in Eq. (4.3). Using a non-zero trend is optional but it is often preferred in practice (see *e.g.* Rasmussen and Williams (2006); Schöbi et al. (2015)). Note that the mean of the Kriging predictor in Eq. (4.8) is not confined to be zero when the trend is zero.

In the literature, it is customary to distinguish between Kriging metamodels depending on the type of trend they use (Stein, 1999; Santner et al., 2003; Rasmussen and Williams, 2006). The most general and flexible formulation is *universal Kriging*, which assumes that the trend is composed of a sum of P arbitrary functions $f_k(\mathbf{x})$, *i.e.*

$$\boldsymbol{\beta}^\top \boldsymbol{f}(\mathbf{x}) = \sum_{k=1}^{P} \beta_k f_k(\mathbf{x}). \tag{4.15}$$

Some of the most commonly used trends for universal Kriging are given for reference in Table 4.1. *Simple Kriging* assumes that the trend has a known

constant value, *i.e.* $P = 1$, $f_1(\mathbf{x}) = 1$ and β_1 is known. In *Ordinary Kriging* the trend has a constant but unknown value, *i.e.* $P = 1$, $f_1(\mathbf{x}) = 1$ and β_1 is unknown.

TABLE 4.1: Formulas of the most commonly used Kriging trends.

Trend	**Formula**
constant (ordinary Kriging)	β_0
linear	$\beta_0 + \sum_{i=1}^{M} \beta_i x_i$
quadratic	$\beta_0 + \sum_{i=1}^{M} \beta_i x_i + \Sigma_{i=1}^{M} \Sigma_{j=1}^{M} \beta_{ij} x_i x_j$

4.2.2 *Correlation function*

The correlation function (also called *kernel* in the literature, or *covariance* function if it includes the Gaussian process variance σ^2) is a crucial ingredient for a Kriging metamodel, since it contains the assumptions about the function that is being approximated. An arbitrary function of $(\boldsymbol{x}, \boldsymbol{x}')$ is in general not a valid correlation function. In order to be admissible, it has to be chosen in the set of positive definite kernels. However, checking for positive definiteness of a kernel can be a challenging task. Therefore it is usually the case in practice to select families of kernels known to be positive definite and to estimate their parameters based on the available experimental design and model responses (see Section 4.2.3).

A usual assumption is to consider kernels that depend only on the quantity $h = \|\boldsymbol{x} - \boldsymbol{x}'\|$ which are called *stationary*. A list of stationary kernels commonly used in the literature can be found in Table 4.2. Different correlation families result in different levels of smoothness for the associated Gaussian processes, as depicted in Figure 4.2 (Rasmussen and Williams, 2006).

In case of multidimensional inputs ($M > 1$), it is common practice to obtain admissible kernels as functions of one-dimensional correlation families as the ones in Table 4.2. Two standard approaches in the literature are the *separable* correlation type (Sacks et al., 1989):

$$R(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}) = \prod_{i=1}^{M} R(x_i, x_i', \theta_i)\,, \tag{4.16}$$

TABLE 4.2: Commonly used correlation families.

Name	Formula
Linear	$R(h;\theta) = \max\left(0, 1 - \frac{\lvert h\rvert}{\theta}\right)$
Exponential	$R(h;\theta) = \exp\left(-\frac{\lvert h\rvert}{\theta}\right)$
Matérn 3/2	$R(h;\theta) = \left(1 + \frac{\sqrt{3}\lvert h\rvert}{\theta}\right)\exp\left(-\frac{\sqrt{3}\lvert h\rvert}{\theta}\right)$
Matérn 5/2	$R(h;\theta) = \left(1 + \frac{\sqrt{5}\lvert h\rvert}{\theta} + \frac{5h^2}{3\theta^2}\right)\exp\left(-\frac{\sqrt{5}\lvert h\rvert}{\theta}\right)$
Gaussian (squared exponential)	$R(h;\theta) = \exp\left(-\sum_{i=1}^{M}\left(\frac{h}{\theta}\right)^2\right)$

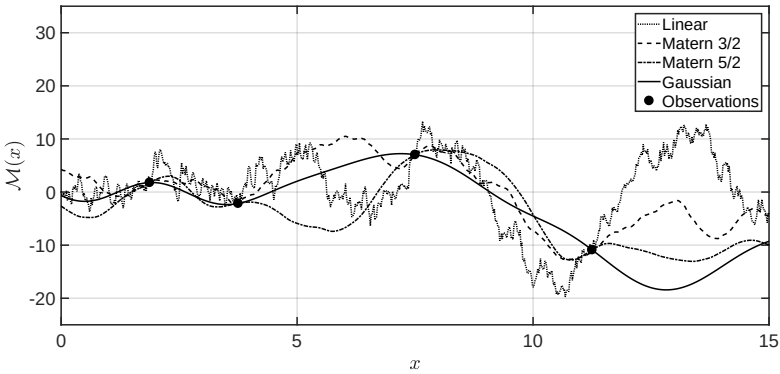


FIGURE 4.2: Realisations of Gaussian processes, characterised by various correlation families and the same length-scale (θ) value.

and the *ellipsoidal* type (Rasmussen and Williams, 2006):

$$R(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}) = R(h)\ ,\ h = \sqrt{\sum_{i=1}^{M}\left(\frac{x_i - x_i'}{\theta_i}\right)^2}\ . \tag{4.17}$$

Although typically $\boldsymbol{\theta} \in \mathbb{R}^M$ this is not necessarily true in the general case, since the number of components of $\boldsymbol{\theta}$ that correspond to each input dimension may vary. In the current stage, it is assumed however that one element of $\boldsymbol{\theta}$ is used per dimension for notational clarity.

In certain scenarios (*e.g.* based on prior knowledge), *isotropic* correlation functions can be used for multidimensional inputs. In that case the same

correlation function parameters $\theta_i \equiv \theta$ are used for each input dimension $i = 1, \ldots, M$ in Eq. (4.16) and Eq. (4.17).

4.2.3 *Estimating the hyperparameters*

In most practical applications of Kriging surrogate modelling, the hyperparameters $\boldsymbol{\theta}$ are estimated given an experimental design $\mathcal{X}$ and model responses $\boldsymbol{y}$. *Maximum likelihood* and *cross-validation* are the most commonly used methods for doing so and further discussed next.

The *maximum likelihood* approach aims at finding the set of parameters $\boldsymbol{\beta}, \boldsymbol{\theta}, \sigma^2$ such that the likelihood of the observations $\boldsymbol{y} = \{\mathcal{M}(\mathbf{x}_1), \ldots, \mathcal{M}(\mathbf{x}_N)\}^\top$ is maximal. Since $\boldsymbol{y}$ follows a multivariate Gaussian distribution, the likelihood function reads:

$$L(\boldsymbol{y} \mid \boldsymbol{\beta}, \sigma^2, \boldsymbol{\theta}) = \frac{\det(\boldsymbol{R})^{-1/2}}{(2\pi\sigma^2)^{N/2}} \exp\left[-\frac{1}{2\sigma^2}(\boldsymbol{y} - \boldsymbol{F}\boldsymbol{\beta})^\top \boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{F}\boldsymbol{\beta})\right]. \quad (4.18)$$

For any given value of $\boldsymbol{\theta}$, the maximisation of the likelihood *w.r.t.* $\boldsymbol{\beta}$ and σ^2 is a convex quadratic programming problem. Consequently, it admits closed form generalized least-squares estimates of $\boldsymbol{\beta}$ and σ^2 (for proof and more details see *e.g.* Santner et al. (2003)):

$$\boldsymbol{\beta} = \boldsymbol{\beta}(\boldsymbol{\theta}) = \left(\boldsymbol{F}^\top \boldsymbol{R}^{-1} \boldsymbol{F}\right)^{-1} \boldsymbol{F}^\top \boldsymbol{R}^{-1} \boldsymbol{y}\,, \quad (4.19)$$

$$\sigma^2 = \sigma^2(\boldsymbol{\theta}) = \frac{1}{N}(\boldsymbol{y} - \boldsymbol{F}\boldsymbol{\beta})^\top \boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{F}\boldsymbol{\beta})\,. \quad (4.20)$$

The value of the hyperparameters $\boldsymbol{\theta}$ is calculated by solving the optimisation problem:

$$\boldsymbol{\theta} = \arg\min_{\mathcal{D}_{\boldsymbol{\theta}}} \left(-\log L(\boldsymbol{y} \mid \boldsymbol{\beta}, \sigma^2, \boldsymbol{\theta})\right). \quad (4.21)$$

Based on Eqs (4.18) - (4.20) the optimisation problem in Eq. (4.21) can be written as follows:

$$\boldsymbol{\theta} = \arg \min_{\mathcal{D}_{\boldsymbol{\theta}}} \left(\frac{1}{2} \log(\det(\boldsymbol{R})) + \frac{N}{2} \log(2\pi\sigma^2) + \frac{N}{2} \right). \tag{4.22}$$

The *cross-validation* method (also known as K-fold cross-validation) is based instead on partitioning the whole set of observations $\mathcal{S} \stackrel{\text{def}}{=} \{\mathcal{X}, \boldsymbol{y}\}$ into K mutually exclusive and collectively exhaustive subsets $\{\mathcal{S}_k\,,\, k = 1, \ldots, K\}$ such that

$$\mathcal{S}_i \cap \mathcal{S}_j = \emptyset\,,\ \forall (i \neq j) \in \{1, \ldots, K\}^2 \ \text{ and } \bigcup_{k=1}^{K} \mathcal{S}_k = \mathcal{S}. \tag{4.23}$$

The k-th set of cross-validated predictions is obtained by calculating the Kriging predictor using all the subsets but the k-th one and evaluating its predictions on that specific k-th fold that was left apart. The leave-one-out cross-validation procedure corresponds to the special case that the number of classes is equal to the number of observations ($K = N$).

In the latter case the objective function is (Santner et al., 2003; Bachoc, 2013):

$$\boldsymbol{\theta} = \arg \min_{\mathcal{D}_{\boldsymbol{\theta}}} \sum_{i=1}^{K} \left(\mathcal{M}(\boldsymbol{x}^{(i)}) - \mu_{\widehat{Y},(-i)}(\boldsymbol{x}^{(i)}) \right)^2, \tag{4.24}$$

where $\mu_{\widehat{Y},(-i)}(\boldsymbol{x}^{(i)})$ is the mean Kriging predictor that was calculated using $\mathcal{S} \setminus \left\{ \boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)} \right\}$, evaluated at $\boldsymbol{x}^{(i)}$. Notice that for the case of leave-one-out cross-validation, i is an index but in the general case i is a vector of indices. Calculating the objective function in Eq. (4.24) would require the calculation of K Kriging surrogates. The computational requirements for performing this operation can be significantly reduced as shown in Dubrule (1983), since an analytical expression of the objective function of Eq. (4.24) is available based on a single Kriging model built with $\mathcal{S}$.

The estimate of σ^2 is calculated as follows (Cressie, 1993; Bachoc, 2013):

$$\sigma^2 = \sigma^2(\boldsymbol{\theta}) = \frac{1}{K}\sum_{i=1}^{K}\frac{\left(\mathcal{M}(\boldsymbol{x}^{(i)}) - \mu_{\widehat{Y},(-i)}(\boldsymbol{x}^{(i)})\right)^2}{\sigma^2_{\widehat{Y},(-i)}(\boldsymbol{x}^{(i)})}, \tag{4.25}$$

where $\sigma^2_{\widehat{Y},(-i)}(\boldsymbol{x}^{(i)})$ denotes the variance of a Kriging predictor that was calculated using $\mathcal{S} \setminus \left\{\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\right\}$, evaluated at point $\boldsymbol{x}^{(i)}$. When i is a set of indices, the division and the squared operations in Eq. (4.25) are performed element-wise.

In this work we mainly use cross-validation for estimating the correlation parameters instead of the maximum likelihood method. This is motivated by the comparative study in Bachoc (2013) between those. The CV method is expected to perform better in cases that the correlation family of the Kriging surrogate is not identical to the one of the true model. This is a common occurrence in engineering practice, especially within a data-driven context which is the focus of this thesis.

Numerically solving the optimisation problems described in Eq. (4.22) (maximum likelihood case) or Eq. (4.24) (cross-validation case) relies on either local (*e.g.* gradient-based) or global (*e.g.* evolutionary) algorithms. On the one hand, local methods tend to converge faster and require fewer objective function evaluations than their global counterparts. On the other hand, the existence of flat regions and multiple local minima, especially for larger input dimension, can lead gradient methods to poor performance when compared to global methods. It is common practice to combine both strategies sequentially to improve global optimisation results with a final local search (which is also known as *hybrid* methods).

4.3 POLYNOMIAL CHAOS EXPANSIONS

4.3.1 *Definition*

Polynomial chaos expansions (PCE) represent a class of surrogate models that has seen widespread use in the context of uncertainty quantification due to their flexibility and efficiency. Consider that $\boldsymbol{X} \in \mathbb{R}^M$ is a random vector with independent components described by the joint PDF $f_{\boldsymbol{X}}$ and that the model output Y in Eq. (4.1) has finite variance. Then the polynomial

chaos expansion of $\mathcal{M}(\boldsymbol{X})$ is given by (Ghanem and Spanos, 1991; Soize and Ghanem, 2004):

$$Y = \mathcal{M}(\boldsymbol{X}) = \sum_{\boldsymbol{\alpha} \in \mathbb{N}^M} \theta_{\boldsymbol{\alpha}} \Psi_{\boldsymbol{\alpha}}(\boldsymbol{X}) \tag{4.26}$$

where the $\Psi_{\boldsymbol{\alpha}}(\boldsymbol{X})$ are multivariate polynomials orthonormal with respect to $f_{\boldsymbol{X}}$, $\boldsymbol{\alpha} \in \mathbb{N}^M$ is a multi-index that identifies the components of the multivariate polynomials $\Psi_{\boldsymbol{\alpha}}$ and the $\theta_{\boldsymbol{\alpha}} \in \mathbb{R}$ are the corresponding coefficients.

To construct the polynomial basis $\Psi_{\boldsymbol{\alpha}}(\boldsymbol{X})$ in Eq. (4.30) we start from a set of *univariate orthonormal polynomials* $\phi_k^{(i)}(x_i)$ which satisfy:

$$\left\langle \phi_j^{(i)}, \phi_k^{(i)} \right\rangle \overset{\text{def}}{=} \int_{\mathcal{D}_{X_i}} \phi_j^{(i)}(x_i) \phi_k^{(i)}(x_i) f_{X_i}(x_i) \mathrm{d}x_i = \delta_{jk} \tag{4.27}$$

where i identifies the input variable w.r.t. which they are orthogonal, as well as the corresponding polynomial family, j and k the corresponding polynomial degree, $f_{X_i}(x_i)$ is the i^{th}-input marginal distribution and δ_{jk} is the Kronecker symbol. Note that this definition of inner product can be interpreted as the expectation value of the product of the multiplicands:

$$\left\langle \phi_j^{(i)}, \phi_k^{(i)} \right\rangle = \mathbb{E}_{X_i} \left[\phi_j^{(i)}(x_i)\, \phi_k^{(i)}(x_i) \right] . \tag{4.28}$$

TABLE 4.3: List of classical univariate polynomial families common in polynomial chaos expansion applications (Sudret, 2007).

Type of variable	Distribution	Orthogonal polynomials	$\psi_k(x)$
Uniform	$\mathbf{1}_{]-1,1[}(x)/2$	Legendre $P_k(x)$	$P_k(x)/\sqrt{\frac{1}{2k+1}}$
Gaussian	$\frac{1}{\sqrt{2\pi}} e^{-x^2/2}$	Hermite $H_{e_k}(x)$	$H_{e_k}(x)/\sqrt{k!}$
Gamma	$x^a\, e^{-x}\, \mathbf{1}_{\mathbb{R}^+}(x)$	Laguerre $L_k^a(x)$	$L_k^a(x)/\sqrt{\frac{\Gamma(k+a+1)}{k!}}$
Beta	$\mathbf{1}_{]-1,1[}(x)\, \frac{(1-x)^a(1+x)^b}{B(a)\, B(b)}$	Jacobi $J_k^{a,b}(x)$	$J_k^{a,b}(x)/\mathfrak{J}_{a,b,k}$
		$\mathfrak{J}_{a,b,k}^2 = \frac{2^{a+b+1}}{2k+a+b+1} \frac{\Gamma(k+a+1)\Gamma(k+b+1)}{\Gamma(k+a+b+1)\Gamma(k+1)}$	

The multivariate polynomials $\Psi_{\boldsymbol{\alpha}}(\boldsymbol{X})$ are then assembled as the tensor product of their univariate counterparts:

$$\Psi_{\boldsymbol{\alpha}}(\boldsymbol{x}) \stackrel{\text{def}}{=} \prod_{i=1}^{M} \phi_{\alpha_i}^{(i)}(x_i)\,. \tag{4.29}$$

For standard distributions, the associated families of orthogonal polynomials are well-known (Xiu and Karniadakis, 2002). Such cases are shown in Table 4.3. Moreover, it is possible to construct polynomials orthogonal w.r.t. any distribution by means of Gram-Schmidt orthonormalisation, a.k.a. Stieltjes procedure for polynomials (Gautschi, 2004).

4.3.2 *Truncation schemes*

In practice, the series in Eq. (4.26) is truncated to a finite sum, by introducing the truncated polynomial chaos expansion:

$$\mathcal{M}(\boldsymbol{X}) \approx \widehat{\mathcal{M}}(\boldsymbol{X}) = \sum_{\boldsymbol{\alpha}\in\mathcal{A}} \theta_{\boldsymbol{\alpha}} \Psi_{\boldsymbol{\alpha}}(\boldsymbol{X}) \equiv \boldsymbol{\theta}^{\top}\boldsymbol{\Psi}(\mathbf{x})\,, \tag{4.30}$$

where $\mathcal{A} \subset \mathbb{N}^M$ is the set of selected multi-indices of multivariate polynomials.

A typical truncation scheme consists in selecting multivariate polynomials up to a total degree p, *i.e.*:

$$\mathcal{A} = \left\{ \boldsymbol{\alpha} \in \mathbb{N}^M \,:\, \|\boldsymbol{\alpha}\|_1 \leq p \right\}, \tag{4.31}$$

where $\|\boldsymbol{\alpha}\|_1 = \sum_{i=1}^{M} \alpha_i$. However, the corresponding number of terms in the truncated series rapidly increases with M, giving rise to the "curse of dimensionality".

To limit the number of basis terms that include higher-order interactions between input variables, which are usually less significant, Blatman (2009) proposed the use of a hyperbolic truncation scheme, which is adopted in this work. In the latter, the set of retained multi-indices is defined as:

$$\mathcal{A} = \left\{ \boldsymbol{\alpha} \in \mathbb{N}^M \,:\, \| \boldsymbol{\alpha} \|_q \leq p \right\}, \tag{4.32}$$

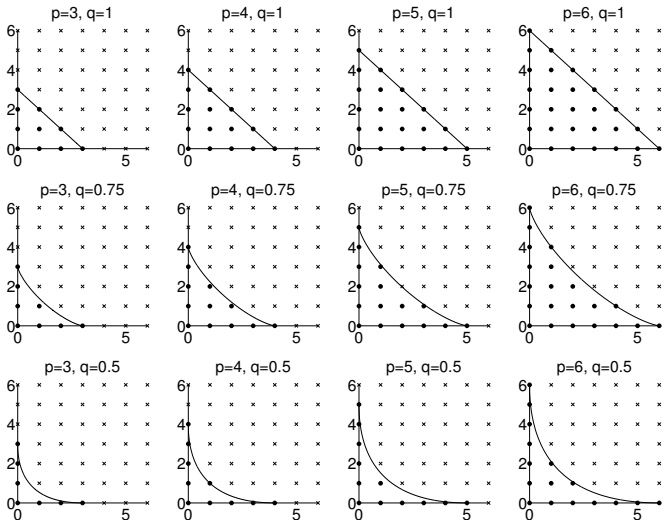


FIGURE 4.3: Visualisation of the hyperbolic truncation scheme for varying values of the polynomial degree p and the q value of the q-norm in Eq. (4.33).

with:

$$\| \boldsymbol{\alpha} \|_q = \left(\sum_{i=1}^{M} \alpha_i^q \right)^{1/q} , \; 0 < q \leq 1 \,, \tag{4.33}$$

where $0 < q \leq 1$ is a parameter and p is the maximal total degree of the retained polynomials. Based on Eq. (4.33), lower values of q correspond to a smaller number of interaction terms in the PCE basis. Notice that for $q = 1$ the hyperbolic truncation scheme is identical to the standard one in Eq. (4.31) and for $q \to 0$, the expansion becomes additive, *i.e.* a sum of univariate functions in the X_i's.

Figure 4.3 visualises the hyperbolic truncation scheme, considering a two-dimensional space ($M = 2$) and varying values of q (constant in each row) and p (constant in each column). In each panel, corresponding to different q and p values, the retained indices in $\mathcal{A}$ are the ones that fall on the curve or lie below it. Overall, this scheme includes all the high-degree terms in each variable, but favours the rejection of higher order interaction terms as q decreases.

4.3.3 *Calculation of the PCE coefficients*

Having specified the truncated polynomial basis terms, calibrating a PCE surrogate refers to the calculation of the expansion coefficients $\theta_{\boldsymbol{\alpha}}$, $\boldsymbol{\alpha} \in \mathcal{A} \subset \mathbb{N}^M$ in Eq. (4.30). We focus on non-intrusive methods to achieve this, *i.e.* by processing the samples $\{\mathcal{X}, \mathcal{Y}\}$. Such techniques can be categorised into three groups: (i) projection (Keese and Matthies, 2005; Le Maître et al., 2002; Ghiocel and Ghanem, 2002), (ii) stochastic collocation (Xiu and Karniadakis, 2002; Xiu and Hesthaven, 2005), and (iii) least-square minimisation (Berveiller et al., 2006; Blatman and Sudret, 2010, 2011; Chkifa et al., 2015) which is the focus in this thesis.

The common aspect in least-square minimisation techniques, is that they calculate $\boldsymbol{\theta}$ by minimising the expectation of the least-squares residual (Berveiller et al., 2006):

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \mathbb{R}^{n_{\mathcal{A}}}}{\arg\min}\, \mathbb{E}\left[\left(\boldsymbol{\theta}^{\mathsf{T}}\Psi(\boldsymbol{X}) - \mathcal{Y}\right)^2\right], \tag{4.34}$$

where the expected value is approximated in practice by:

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \mathbb{R}^{n_{\mathcal{A}}}}{\arg\min} \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{\boldsymbol{\alpha} \in \mathcal{A}} \theta_{\boldsymbol{\alpha}} \psi_{\boldsymbol{\alpha}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2. \tag{4.35}$$

A direct solution of Eq. (4.35) is given by Ordinary Least-Squares (OLS). Given the samples $\{\mathcal{X}, \mathcal{Y}\}$, the OLS solution of Eq. (4.34) reads:

$$\widehat{\boldsymbol{\theta}} = (\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\mathsf{T}}\mathcal{Y}, \tag{4.36}$$

where

$$A_{ij} = \Psi_j\left(\mathbf{x}^{(i)}\right), i = 1, \ldots, N, j = 0, \ldots, P-1, \tag{4.37}$$

is the so-called *experimental* (or *information*) *matrix*.

In applied science problems, only low order interactions between the input variables tend to be important. This is a common occurrence known as

sparsity-of-effects principle (Wu and Hamada, 2000). Truncation schemes such as the hyperbolic truncation in Eq. (4.33) assist in producing sparse PCEs. A complementary strategy to favour sparsity in high dimension is based on a regularised version of the least-squares problem in Eq. (4.35):

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\mathbb{R}^{n_{\mathcal{A}}}}{\arg\min}\, \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{\boldsymbol{\alpha}\in\mathcal{A}} \theta_{\boldsymbol{\alpha}}\psi_{\boldsymbol{\alpha}}(\mathbf{x}^{(i)}) - y^{(i)}\right)^2 + \lambda\left\|\boldsymbol{\theta}\right\|_1 . \tag{4.38}$$

The regularisation term $\|\boldsymbol{\theta}\|_1 = \sum_{\boldsymbol{\alpha}\in\mathcal{A}} |\theta_{\boldsymbol{\alpha}}|$ penalises non-sparse solutions with a magnitude that is controlled by the regularisation coefficient λ. Several algorithms exist that solve the minimisation problem in Eq. (4.38), including least absolute shrinkage and selection operator (LASSO, Tibshirani, 1996), forward stage-wise regression (Hastie et al., 2007) and least-angle regression (LAR, Efron et al. (2004)). In this work we adopt the LAR algorithm implementation that is proposed by Blatman and Sudret (2011).

4.3.4 *Dealing with arbitrary probabilistic input spaces*

Although the general formulation of PCE in Eq. (4.26) assumes independent random input variables, it is possible to extend it to more complex cases where the independence assumption does not hold or no standard polynomials are defined for the marginal distributions of $\mathbf{X}$. There exists an isoprobabilistic transformation $\mathcal{T}$ of $\mathbf{X} \sim f_{\mathbf{X}}(\mathbf{x})$ such that:

$$\mathbf{Z} = \mathcal{T}(\mathbf{X}), \quad \mathbf{X} = \mathcal{T}^{-1}(\mathbf{Z}), \tag{4.39}$$

where $\mathbf{Z}$ is a random vector with independent components distributed according to one of the distributions in Table 4.3. Hence, we can rewrite Eq. (4.26) as follows:

$$Y = \mathcal{M}(\boldsymbol{X}) = \sum_{\boldsymbol{\alpha}\in\mathbb{N}^M} \theta_{\boldsymbol{\alpha}}\, \Psi_{\boldsymbol{\alpha}}(\mathcal{T}(\boldsymbol{X})). \tag{4.40}$$

This also allows to use any type of orthonormal polynomial with any type of input marginals at the cost of an additional transform. However, this type of transform can be highly non-linear, which may lead to reduced accuracy of the truncated PCE. This is relevant especially when transforming between distributions with compact and non-compact support (*e.g.*

uniform to Gaussian). The topic of the isoprobabilistic transformation, and its implementation when the dependence is described using the copula formalism, is further discussed in Section 6.2.3.

4.4 ERROR MEASURES

The calculation of any surrogate model is typically followed by the assessment of its predictive performance. Several error measures are available to provide such a quantitative assessment. Arguably the most well-known accuracy measure for most surrogates is the relative generalisation error ε_{gen} that reads:

$$\varepsilon_{gen} = \mathbb{E}\left[\left(Y - \widehat{\mathcal{M}}(\mathbf{X};\boldsymbol{\theta})\right)^2\right] / \mathrm{Var}\left[Y\right]. \tag{4.41}$$

This error measure (or, more precisely, one of its estimators) is also the ideal objective function for the optimisation process involved in the calibration of the surrogate parameters $\boldsymbol{\theta}$. In most practical situations, however, it is not possible to calculate ε_{gen} analytically. An estimator $\widehat{\varepsilon}_{gen}$ of this error can be computed by comparing the true and surrogate model responses evaluated at a sufficiently large *validation set* $\mathcal{X}_v = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N_v)}\right\}$ of size N_v:

$$\widehat{\varepsilon}_{gen} = \frac{\sum_{i=1}^{N_v}\left(\mathcal{M}(\mathbf{x}^{(i)}) - \widehat{\mathcal{M}}(\mathbf{x}^{(i)})\right)^2}{\sum_{i=1}^{N_v}\left(\mathcal{M}(\mathbf{x}^{(i)}) - \widehat{\mu}_Y\right)^2}, \tag{4.42}$$

where $\widehat{\mu}_Y = \frac{1}{N}\sum_{i=1}^{N_v}\mathcal{M}(\mathbf{x}^{(i)})$ is the sample mean of the validation set responses and $\widehat{\mathcal{M}}(\mathbf{x}^{(i)})$ is used in place of $\widehat{\mathcal{M}}(\mathbf{x}^{(i)};\boldsymbol{\theta})$ to simplify the notation.

In data-driven applications, or when the computational model is expensive to evaluate, only a single set $\mathcal{S} \overset{\text{def}}{=} \{\mathcal{X}, \mathcal{Y}\}$ is often available. The entire set is therefore used for calculating the surrogate parameters. Estimating the generalisation error by means of Eq. (4.42) on the same set, however, corresponds to computing the so-called *empirical error*, which is prone to underestimate drastically the true generalisation error, due to the overfitting phenomenon. In such cases, a fair approximation of $\widehat{\varepsilon}_{gen}$ can be obtained by means of cross-validation (CV) techniques (see *e.g.* Hastie et al., 2001).

In k-fold CV, $\mathcal{S}$ is randomly partitioned into k mutually exclusive and collectively exhaustive sets $\mathcal{S}_i$ of approximately equal size:

$$\mathcal{S}_i \cap \mathcal{S}_j = \emptyset \ , \ \forall (i \neq j) \in \{1, \ldots, k\}^2 \ \text{ and } \bigcup_{i=1}^{k} \mathcal{S}_i = \mathcal{S}. \tag{4.43}$$

The k-fold cross-validation error ε_{CV} reads:

$$\varepsilon_{CV} = \frac{\sum_{i=1}^{k} \sum_{\mathbf{x} \in \mathcal{S}_i} \left(\mathcal{M}(\mathbf{x}) - \widehat{\mathcal{M}}^{\mathcal{S} \setminus \mathcal{S}_i}(\mathbf{x}) \right)^2}{\sum_{\mathbf{x} \in \mathcal{S}} \left(\mathcal{M}(\mathbf{x}) - \widehat{\mu}_Y \right)^2}, \tag{4.44}$$

where $\widehat{\mathcal{M}}^{\mathcal{S}}_{\mathcal{S} \setminus \mathcal{S}_i}$ denotes the surrogate model that is calculated using $\mathcal{S}$ excluding $\mathcal{S}_i$. The bias of the generalisation error estimator is expected to be minimal in the extreme case of *leave-one-out (LOO) cross-validation* (Arlot and Celisse, 2010), which corresponds to $N-$fold cross validation. The LOO error ε_{LOO} is calculated as in Eq. (4.44) after substituting the set S_i by the singleton $\left\{ \mathbf{x}^{(i)} \right\}$ (*i.e.* $k = N$):

$$\varepsilon_{LOO} = \frac{\sum_{i=1}^{N} \left(\mathcal{M}(\mathbf{x}^{(i)}) - \widehat{\mathcal{M}}^{\setminus i}(\mathbf{x}^{(i)}) \right)^2}{\sum_{i=1}^{N} \left(\mathcal{M}(\mathbf{x}^{(i)}) - \widehat{\mu}_Y \right)^2}, \tag{4.45}$$

where the term $\mathcal{M}^{\setminus i}(\mathbf{x}^{(i)})$, denotes the surrogate built from the set $S \setminus \left\{ \mathbf{x}^{(i)} \right\}$, evaluated at $\mathbf{x}^{(i)}$. The calculation of ε_{LOO} seems to be computationally expensive, because it seems to require the evaluation of N surrogates, but it does not require any additional run of the full computational model. As for Kriging (see Section 4.2.3), Blatman and Sudret (2011) show that the ε_{LOO} in Eq. (4.45) of a polynomial chaos expansion (calculated using least-squares minimisation) can be evaluated in closed form from a *single* surrogate $\widehat{\mathcal{M}}^{PC}$ as follows:

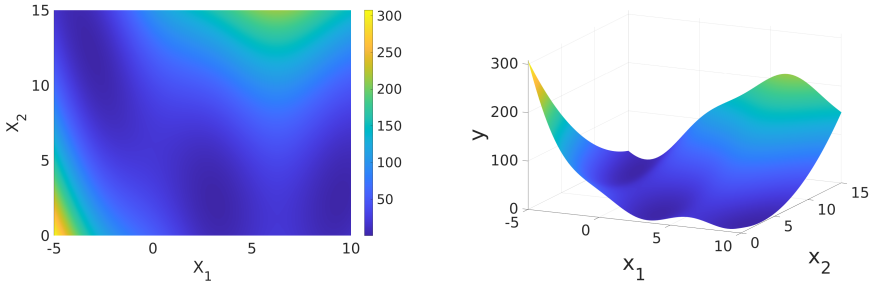


FIGURE 4.4: The Branin-Hoo function visualised in 2D (left) and 3D (right).

$$\varepsilon_{LOO} = \sum_{i=1}^{N} \left(\frac{\mathcal{M}(\boldsymbol{x}^{(i)}) - \widehat{\mathcal{M}}^{PC}(\boldsymbol{x}^{(i)})}{1 - h_i} \right)^2 \Big/ \sum_{i=1}^{N} \left(\mathcal{M}(\boldsymbol{x}^{(i)}) - \widehat{\mu}_Y \right)^2, \tag{4.46}$$

where h_i is the i^{th} component of the vector given by:

$$\boldsymbol{h} = \operatorname{diag}\left(\boldsymbol{A}(\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\mathsf{T}} \right), \tag{4.47}$$

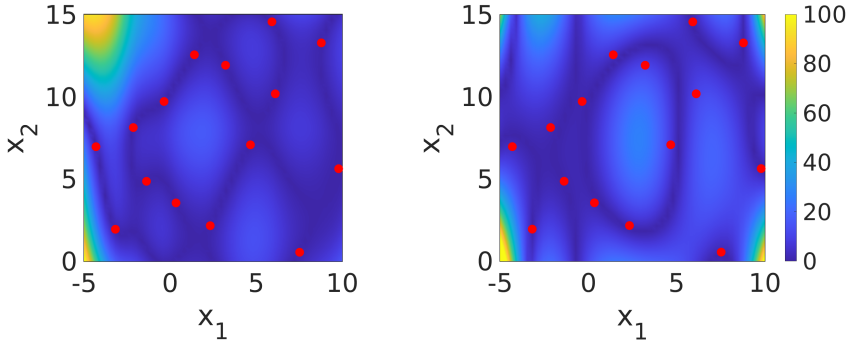
and $\boldsymbol{A}$ is the experimental matrix in Eq. (4.37).

4.5 VISUALISATION

In order to visualise the general behaviour of Kriging and PCE, consider that the computational model is the Branin-Hoo function. It reads (Forrester et al., 2008):

$$\mathcal{M}(\mathbf{x}) = a\left(x_2 - bx_1^2 + cx_1 - r^2\right)^2 + s\,(1-t)\cos(x_1) + s\,, \tag{4.48}$$

where $\mathbf{X} \in \mathbb{R}^2$ and some standard values of the parameters are used, namely $a = 1$, $b = 5.1(4\pi^2)$, $c = 5/\pi$, $r = 6$, $s = 10$ and $t = 1/(8\pi)$. The input random variables are considered independent and uniformly distributed: $X_1 \sim \mathcal{U}(-5, 10)$, $X_2 \sim \mathcal{U}(0, 15)$. A visualisation of the Branin-Hoo function is shown in Figure 4.4.



(a) Kriging residual

(b) PCE residual

FIGURE 4.5: Graphical comparison of the residual $|\mathcal{M}(\mathbf{X}) - \widehat{\mathcal{M}}(\mathbf{X})|$ using either a Kriging or a PCE surrogate.

The goal is to construct a surrogate using each technique based on a limited number of observations and compare the resulting performance. The surrogates are deployed using the UQLAB software framework (Marelli and Sudret, 2014) and their implementation therein using the respective modules (Marelli and Sudret, 2018; Lataniotis et al., 2018).

The Kriging surrogate is constructed using an ellipsoidal correlation function (see Eq. (4.17)) that uses the Matérn 5/2 family (see Table 4.2). We perform ordinary Kriging, *i.e.* the trend in Eq. (4.3) reduces to a single unknown β_0. The hyperparameters are estimated by minimising the CV objective function in Eq. (4.24) using a hybrid genetic algorithm.

Next, we construct a sparse PCE with maximal polynomial degree $p = 10$ using the degree-adaptive LARS algorithm. In both cases the same dataset $\mathcal{S}$ with $N = 15$ random samples is used to calibrate the surrogate. An additional set $\mathcal{S}_v = \{\mathcal{X}_v, \mathcal{Y}_v\}$ that corresponds to the observations on a 50×50 regular grid of the input domain (2,500 samples in total) is used for validation purposes.

Figure 4.5 showcases the residual $|\mathcal{M}(\mathbf{X}) - \widehat{\mathcal{M}}(\mathbf{X})|$ evaluated on $\mathcal{S}_v$, where $\widehat{\mathcal{M}}(\mathbf{X})$ corresponds to the Kriging (resp. PCE) surrogate in Figure 4.5a (resp. Figure 4.5b). The experimental design is also shown as red dots. In addition, the value of different error measures related to the predictive performance

of each surrogate are listed in Table 4.4. The LOO error ε_{LOO}, is calculated on the experimental design as described in Eq. (4.45). The generalisation error $\widehat{\varepsilon}_{gen}$ is estimated from the validation set instead (Eq. (4.42)).

TABLE 4.4: Error measures of the Kriging and PCE surrogate of the Branin Hoo function (experimental design of size 15).

Error measure	Kriging	PCE
ε_{loo}	0.0808	0.0709
$\widehat{\varepsilon}_{gen}$	0.1252	0.1061

At a first glance, both techniques show comparable performance based on the prediction error estimators in Table 4.4 but there are some notable differences between the residual landscape of each method. Overall, we observe that the Kriging surrogate achieves low residual values especially in the vicinity of the experimental design samples. This is indeed expected due to its interpolating property. Although the PCE surrogate shows increased residual values around some of the experimental design samples, it achieves a lower generalisation error, as seen from its estimator $\widehat{\varepsilon}_{gen}$. The improvement of the PCE compared to Kriging becomes more significant in regions outside the given observations, *i.e.* when extrapolating.

4.6 CONCLUSION

Surrogate models are characterised by their ability to emulate complex computational models based on a relatively small set of model runs used for training and by their inexpensive evaluation. This makes them an appealing tool not only for substituting a computational model that is costly to run but also within data-driven applications where no such model is available.

Regardless of their specificity, surrogate models can be summarised as a function of the form $Y = \widehat{\mathcal{M}}(\mathbf{X}; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ denotes their parameters. Calculating (or calibrating) a surrogate refers to the estimation of $\boldsymbol{\theta}$. It corresponds to the minimiser of an objective function which is typically an estimator of the generalisation error of the surrogate.

Two surrogate modelling techniques that are used extensively in this thesis, namely Kriging and polynomial chaos expansions, were described in more

detail. Finally, we showcased their performance in a benchmark problem and highlighted some key differences between them.

5

SURROGATE MODELLING WITH HIGH-DIMENSIONAL INPUTS

5.1 INTRODUCTION

Surrogate models have become staple tools in the arsenal of uncertainty quantification, thanks to their versatility, ease of deployment and high-performance. Parametrising and training a surrogate model, however, can become harder or even intractable as the number of input parameters increases, a well known problem often referred to as *curse of dimensionality* (see *e.g.* Verleysen and François, 2005).

For the sake of clarity, in the following we will classify high-dimensional inputs in two broad categories, depending on their characteristics: *unstructured* and *structured*. Unstructured inputs are characterised by the lack of an intrinsic ordering, and they are commonly identified with the so-called "model parameters", *e.g.* point loads on mechanical models, or resistance values in electrical circuit models. Structured inputs, on the other hand, are characterised by the existence of a natural ordering and/or a distance function (*i.e.* they show strong correlation across some physically meaningful set of coordinates), as it is typical for time-series or space-variant quantities. Boundary conditions in complex simulations that rely on discretisation grids, *e.g.* time-dependent excitations at grid nodes, often belong to this second class. In most practical applications, unstructured inputs range in dimension in the order $\mathcal{O}(10^{0-2})$, while structured inputs tend to be in the order $\mathcal{O}(10^{2-6})$.

Several strategies have been explored in the literature to deal with high dimensional problems for surrogate modelling (SM). A common approach in dealing with unstructured inputs is input variable selection, which consists in identifying the "most important" inputs according to some sensitivity measure, see *e.g.* Iooss and Lemaître (2015a), and simply ignoring the others (*e.g.* by setting them to their nominal value).

In the context of kernel-based emulators (*e.g.* Kriging or support vector machines), some attention has been devoted to the use of simple isotropic kernels (Djolonga et al., 2013), or to the design of specific kernels for high-dimensional input vectors, sometimes including deep learning techniques (*e.g.*, Lawrence, 2005; Durrande et al., 2012; Wilson et al., 2016).

The more general concept of *dimensionality reduction* (DR) is applied in more complex scenarios, which essentially consists in mapping the input space into a suitable lower dimensional space using an appropriate transformation (see Chapter 3 for a more detailed introduction). The latter approach is considered in this work due to its applicability to cases for which variable selection seems inadequate or insufficient (*e.g.* in the presence of structured inputs).

In the current literature, a two-step approach is often followed for dealing with such problems: first, the input dimension is reduced; then, the surrogate model is constructed directly in the reduced space. The dimensionality reduction step is therefore based on an *unsupervised* objective, *i.e.* an objective that only takes into account the input observations. Examples of unsupervised objectives include the minimisation of the input reconstruction error (see *e.g.* autoencoders in Section 3.5), maximisation of the sample variance (in PCA, Section 3.2), maximisation of statistical independence (Hyvärinen and Oja, 1997), and preservation of the distances between the observations (such as MDS and variants presented in Section 3.3). While in principle attractive due to their straightforward implementation, unsupervised approaches for dimensionality reduction may be suboptimal in this context, because the input-output map of the reduced representation may exhibit a complex topology unsuitable for surrogate modelling (Wahlström et al., 2015; Calandra et al., 2016).

To deal with this issue, various *supervised* techniques have been proposed, in the sense that the objective of the input compression somehow takes into account the model response. One such approach that has received attention recently is based on the so-called *active subspaces* concept (Constantine et al., 2014). Various methods that belong to this category provide a linear transformation of the high dimensional input space into a reduced space that is characterised by maximal variability w.r.t. the model output. However, active subspaces methods often require the availability of the model gradient w.r.t. the input parameters, a limiting factor in data-driven scenarios where such information is not available and needs to be approximated (For-

nasier et al., 2012). Moreover, the numerical computation of the gradient may be infeasible for problems that involve structured inputs such as time series or multidimensional maps with $\mathcal{O}(10^{2-6})$ components. Yang et al. (2018) propose a technique to compress the input space for more efficient application of polynomial chaos expansions with Hermite polynomials. This work is closely tied to a specific surrogate modelling technique and assumes certain statistical properties of the input space (Gaussian random variables), with the possibility however to expand this method to other input distributions.

Other data-driven supervised DR techniques have been proposed in the literature, which are dependent on the properties of a specific combination of either DR or SM techniques. Hinton and Salakhutdinov (2006) employ multi-layer neural networks for both the DR and the SM steps. Specifically, an unsupervised objective based on the reconstruction error is followed by a generalisation performance objective that aims at fine tuning the network weights with respect to a measure of the surrogate modelling error. Similar approaches have been proposed with other combinations of methods. In Damianou and Lawrence (2013), the same idea is extended by using stacked Gaussian processes instead of multilayer neural networks. In Huang et al. (2015); Calandra et al. (2016) this approach is extended by combining neural networks with Gaussian processes within a Bayesian framework.

All these methods demonstrate that supervised approaches yield a significant accuracy advantage over the unsupervised ones, as the final goal of the supervised learner (*i.e.* surrogate model accuracy) matches the final goal of high-dimensional surrogate modelling in the first place. However, this increased accuracy comes at the cost of restricting the applicability of such methods to specific combinations of DR and SM techniques.

In this chapter, we propose a novel method for performing dimensionality reduction together with surrogate modelling, which we name (perhaps with a lack of creative flair) DRSM. The aim of this method is to capitalise on the performance gains of supervised DR, while maintaining maximum flexibility in terms of both DR and SM methodologies. Recognising that different communities, applications and researchers have in general access to one or two preferred techniques for either DR or SM, the proposed approach is fully non-intrusive, *i.e.* both the DR and the SM stages are considered as *black boxes* under very general conditions. The novelty lies

in the way the two stages are coupled into a single problem, for which dedicated solvers are proposed.

This chapter is structured as follows: as a motivation for the proposed approach, in Section 5.2 the classical approach of sequentially applying DR and SM is investigated on a set of benchmark datasets from the machine learning community. The core framework underlying DRSM is then introduced in Section 5.3. Finally, in Section 5.4 the effectiveness of DRSM is analysed on several benchmark applications including both unstructured and structured inputs.

5.2 MOTIVATION - PRELIMINARY STUDY

During the preliminary stages of this work, the classical sequential application of DR and SM was first considered. The idea was that by using state-of-the-art DR techniques instead of classical ones, such as PCA, the performance of the surrogate may increase together with the potential improvement in the input compression. The goal of this section is to summarise the findings of this preliminary study after formally introducing the problem.

Consider the experimental design $\mathcal{S} = \{\mathcal{X}, \mathcal{Y}\}$ that consists of the observations $\mathcal{X} = \left\{\mathbf{x}^{(i)} \in \mathbf{X} \subseteq \mathbb{R}^M, i = 1, \dots, N\right\}$ and the corresponding, scalar, model responses $\mathcal{Y} = \left\{\mathcal{M}\left(\mathbf{x}^{(1)}\right), \dots, \mathcal{M}\left(\mathbf{x}^{(N)}\right)\right\}$ and assume that it is the only available information about the model under investigation. Moreover, the dimensionality of the input space is high, *i.e.* M is large, say $\mathcal{O}\left(10^{2-4}\right)$. The goal is to calculate a surrogate that serves as an approximation of the real model solely based on the available samples. This is a key ingredient for subsequent analyses in the context of UQ, as will be further discussed in Chapter 6.

To distinguish between various computational schemes, we denote from now on by $\widehat{\mathcal{M}}|\mathcal{X}, \mathcal{Y}$ a surrogate model whose parameters $\boldsymbol{\theta}$ are calculated from the experimental design $\mathcal{X}$ and associated model response $\mathcal{Y}$. Due to the high input dimensionality, a surrogate $\widehat{\mathcal{M}}|\mathcal{X}, \mathcal{Y}$ may lead to poor generalisation performance or it may not even be computationally tractable. To reduce the dimensionality, the class of DR methods was introduced in Chapter 3. A DR transformation, expressed by $\mathcal{Z} = g(\mathcal{X}; \mathbf{w})$, can provide a

compressed experimental design, *i.e.* $\mathbf{z}^{(i)} \in \mathbb{R}^m$, $i = 1, \ldots, N$ with $m \ll M$. The surrogate $\widehat{\mathcal{M}}|\mathcal{Z}, \mathcal{Y}$ becomes tractable if m is sufficiently small.

The sequential application of DR and SM is presented next. It simply refers to a two-stage approach, where the DR transformation $\mathbf{X} \to \mathbf{Z}$ is calculated first, followed by the calculation of the surrogate $\widehat{\mathcal{M}}|\mathcal{Z}, \mathcal{Y}$. As discussed in Section 3.1, performing DR corresponds to finding the optimal parameters $\mathbf{w}$ of the mapping $\mathbf{Z} = g(\mathbf{X}; \mathbf{w})$. Their optimal value $\widehat{\mathbf{w}}$ is determined by minimising a loss function. For usual compression goals, focus is given to a specific loss function that quantifies the compressive performance of $g(\cdot)$ by means of the reconstruction error, which reads:

$$\left\{\widehat{\mathbf{w}}, \widehat{\mathbf{w}}'\right\} = \arg\min_{\mathcal{D}_{\mathbf{w}}} \ell(\mathbf{w}; \mathcal{X}) = \mathbb{E}\left[\left\|\mathbf{X} - \widehat{\mathbf{X}}\right\|^2\right], \tag{5.1}$$

where $\widehat{\mathbf{X}} = g^{-1}(\mathbf{Z}, \mathbf{w}')$ denotes the reconstruction of $\mathbf{X}$, calculated through the inverse transform $g^{-1} : \mathbf{Z} \to \mathbf{X}$. In Eq. (5.1), the general case is considered where the parameters associated to $g(\cdot)$ and $g^{-1}(\cdot)$ differ (see *e.g.* KPCA in Section 3.4).

Calculating the surrogate $\widehat{\mathcal{M}}(\mathbf{Z}; \boldsymbol{\theta})$ refers to the calibration of its hyperparameters $\boldsymbol{\theta}$ based on a performance measure such as the generalisation error (see Section 4.4):

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\mathcal{D}_{\boldsymbol{\theta}}} \widehat{\varepsilon}_{gen}(\boldsymbol{\theta}; \mathcal{S}) = \arg\min_{\mathcal{D}_{\boldsymbol{\theta}}} \mathbb{E}\left[\left(Y - \widehat{\mathcal{M}}(\mathbf{Z}; \boldsymbol{\theta})\right)^2\right] / \mathrm{Var}\left[Y\right]. \tag{5.2}$$

The *sequential application* of DR and SM is investigated by examining the relationship between the compressive performance of DR and the resulting generalisation performance of the SM. Within this setting, various DR techniques are combined with Kriging (Section 4.2) on a set of benchmark datasets. These datasets were obtained by the UCI Machine Learning Repository (Dheeru and Karra Taniskidou, 2017) and have served as benchmarks by several authors in the context of metamodelling. Due to the lack of knowledge regarding the intrinsic dimension, in each case the size of the compressed space was set to roughly the half of the initial one.

We consider the DR techniques from Chapter 3 that provide an inverse transformation g^{-1}, namely PCA (Section 3.2), kernel PCA (Section 3.4) and autoencoders (Section 3.5). The input space is compressed from M to m dimensions for each dataset reported in Table 5.1.

Each dataset is split into two mutually exclusive sets: the experimental design $\mathcal{S}$ that is used for calibrating the DR and SM parameters and the validation set $\mathcal{S}_v = \{\mathcal{X}_v, \mathcal{Y}_v\}$, consisting of N_v samples that is used to calculate the performance metrics. The performance of the input compression is evaluated based on the scaled reconstruction error:

$$\widehat{\varepsilon}_{recon} = \frac{1}{M} \sum_{i=1}^{M} \frac{\sum_{j=1}^{N_v} \left(x_{v,i}^{(j)} - \widehat{x}_j^{(i)} \right)^2}{\sum_{j=1}^{N_v} \left(x_{v,i}^{(j)} - \widehat{\mu}_{v,i} \right)^2}, \tag{5.3}$$

where $x_{v,i}^{(j)}$ denotes the $i-th$ component of the validation set sample $\mathbf{x}_v^{(j)}$, $\widehat{x}_j^{(i)}$ the corresponding component of the reconstructed sample and $\widehat{\mu}_{v,i}$ the sample mean of the i-th component of $\mathbf{x}$ evaluated on $\mathcal{X}_v$. This is a scaled version of the estimator of the reconstruction error in Eq. (5.1). The performance of the Kriging surrogate $\widehat{\mathcal{M}}|\mathcal{Z}, \mathcal{Y}$ is evaluated based on the estimator of the generalisation error in Eq. (5.2):

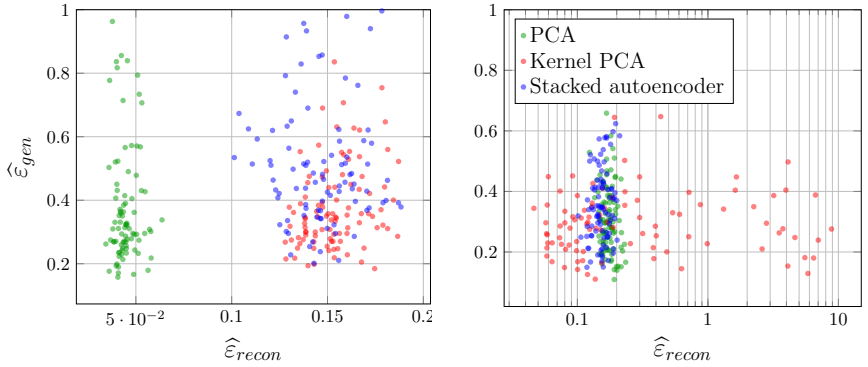
$$\widehat{\varepsilon}_{gen} = \frac{\sum_{i=1}^{N_v} \left(\mathcal{M}(\mathbf{z}_v^{(i)}) - \widehat{\mathcal{M}}(\mathbf{z}_v^{(i)}) \right)^2}{\sum_{i=1}^{N_v} \left(\mathcal{M}(\mathbf{z}_v^{(i)}) - \widehat{\mu}_{y,v} \right)^2}, \tag{5.4}$$

where $\mathcal{M}(\mathbf{z}_v^{(i)})$ simply corresponds to the i-th sample of $\mathcal{Y}_v$, $\widehat{\mathcal{M}}(\mathbf{z}_v^{(i)})$ is the output of the surrogate evaluated on the i-th sample of $\mathcal{Z}_v = g(\mathcal{X}_v; \widehat{\mathbf{w}})$ and $\widehat{\mu}_{y,v}$ denotes the sample mean of $\mathcal{Y}_v$. To evaluate the robustness of the results, the same process is repeated 100 times, each corresponding to a different random split of the dataset into $\mathcal{S}$ and $\mathcal{S}_v$ resulting to different values of $\widehat{\varepsilon}_{recon}$ and $\widehat{\varepsilon}_{gen}$ from Eq. (5.3) and Eq. (5.4), respectively.

The comparison between the input compression and surrogate performance is conducted by comparing their summary statistics in Table 5.1 as well as inspecting their scatter in Figure 5.1.

TABLE 5.1: Results on sequential application of DR and SM on machine learning benchmark datasets

DR method	M	m	N	N_v	$\widehat{\varepsilon}_{recon}$		$\widehat{\varepsilon}_{gen}$	
					$\widehat{\mu}$	$\widehat{\sigma}$	$\widehat{\mu}$	$\widehat{\sigma}$
Car mileage dataset								
PCA	6	3	314	78	0.0447	0.0055	0.5767	0.5888
Kernel PCA					0.1535	0.0148	0.3841	0.1488
Stacked autoencoder					0.1468	0.0188	0.5786	0.3198
Boston housing dataset								
PCA	13	6	456	50	0.1753	0.0208	0.3536	0.3439
Kernel PCA					1.0234	1.8610	0.2976	0.1181
Stacked autoencoder					0.1561	0.0237	0.3772	0.1629



(a) Car mileage dataset

(b) Housing dataset

FIGURE 5.1: Comparison between compression and surrogate model performance using various dimensionality reduction techniques on various datasets.

On the car mileage dataset, PCA achieved the best compressive performance with a mean reconstruction error that is more than an order of magnitude smaller than those obtained by the other methods. However, the best surrogate performance, in terms of the mean generalisation error, was achieved by KPCA, even though it achieved the worst compressive performance. On the housing dataset, non-linear DR techniques can achieve improved

compression compared to PCA, although KPCA shows higher variability, and therefore a poor overall performance over the 100 repetitions. At the same time, the surrogate model performance is comparable in all three cases regardless of the input compression technique.

Overall, from the summary statistics of the reconstruction and generalisation performance metrics in Table 5.1 it is not possible to identify a pattern of dependence or correlation. This is further highlighted by observing their scatter in Figures 5.1a and 5.1b for the car mileage and housing dataset, respectively. There is no clear dependence between the quality of the input compression and the performance of the surrogate.

This can be attributed to the fact that by performing the input compression $\mathbf{X} \rightarrow \mathbf{Z}$, the mapping $\mathbf{Z} \rightarrow Y$ may become less suitable for surrogate modelling regardless of the quality of the compression. However, within the context of this work, the goal is to achieve an optimally performing surrogate. Hence, typical performance measures used for compression such as the reconstruction error, seem unsuitable in this context since they do not take into account the smoothness of the mapping $\mathbf{Z} \rightarrow Y$ which is critical in terms of the performance of any surrogate modelling technique. This observation motivates the proposed original approach for combining DR and SM that is presented in the next section.

5.3 NON-INTRUSIVE SUPERVISED LEARNING: THE PROPOSED DRSM APPROACH

5.3.1 *Introduction*

Consider the setting introduced in Section 5.2, where an experimental design $\mathcal{S} = \{\mathcal{X}, \mathcal{Y}\}$ is available and $\mathbf{X}$ is characterised by large dimensionality. Hence, DR is performed and the surrogate $\widehat{\mathcal{M}}|\mathcal{Z}, \mathcal{Y}$ is computed. The potential of $\widehat{\mathcal{M}}|\mathcal{Z}, \mathcal{Y}$ to achieve satisfactory generalisation performance depends on (i) the learning capacity of the surrogate itself and (ii) the assumption that the input-output map $\mathbf{X} \mapsto Y$ can be sufficiently well approximated by a smaller set of features via the transformation $g(\cdot)$. This discussion focuses on the latter and assumes that the learning capacity of the surrogate is adequate. In case of unstructured inputs, the importance of each input variable may vary depending on the output of interest. In case of structured inputs, there is typically high correlation between the input components.

Hence, in both families of problems a low-dimensional representation may often approximate well the input-output map.

5.3.2 *Problem statement*

The goal of DRSM is to optimise the parameters $\mathbf{w}$ of the compression scheme so that the auxiliary variables $\mathbf{Z} = g(\mathbf{X}; \mathbf{w})$ are suitable to achieve an overall accurate surrogate. Notice that $\mathbf{w}$ typically includes the size of the reduced space, m, provided that it is not known a priori. The general formulation of this problem reads:

$$\left\{\widehat{\mathbf{w}}, \widehat{\boldsymbol{\theta}}\right\} = \underset{\mathbf{w}\in\mathcal{D}_{\mathbf{w}}, \boldsymbol{\theta}\in\mathcal{D}_{\boldsymbol{\theta}}}{\arg\min}\ \ell\left(\mathcal{M}(\cdot), \widehat{\mathcal{M}}\left(g(\cdot; \mathbf{w}), \boldsymbol{\theta}\right)\right), \tag{5.5}$$

where ℓ denotes the loss function that quantifies the generalisation performance of the surrogate. In practice, if a validation set is available, ℓ corresponds to a generalisation error estimator like the one in Eq. (4.42). In the absence of a validation set, then either the LOO estimator in Eq. (4.45) or its k-fold CV counterpart in Eq. (4.44) are used instead. In the following, it is assumed that a validation set is not available and the generalisation error is estimated by the LOO error, hence ℓ is substituted by the ε_{LOO} expression in Eq. (4.45).

5.3.3 *Nested optimisation*

The proposed approach for solving Eq. (5.5), is related to the concept of *block-coordinate descent* (Bertsekas, 1999). During optimisation, the parameters $\mathbf{w}$ and $\boldsymbol{\theta}$ are updated in an alternating fashion. One of the main reasons for this choice is that the optimisation steps of both DR and SM techniques are often tuned ad-hoc to optimise their performance. Examples include sparse linear regression for polynomial chaos expansions (Blatman and Sudret, 2011), or quadratic programming for support vector machines for regression (Vapnik, 1995). A single joint optimisation, albeit potentially yielding accurate results, would require the definition of complex constraints on the different sets of parameters $\mathbf{w}$ and $\boldsymbol{\theta}$, which would be problem- and surrogate-model-

dependent. Therefore, the problem in Eq. (5.5) is expressed as a nested-optimisation problem. The *outer loop optimisation* reads:

$$\widehat{\mathbf{w}} = \underset{\mathbf{w}\in\mathcal{D}_{\mathbf{w}}}{\arg\min}\ \varepsilon_{LOO}(\mathbf{w}; \widehat{\boldsymbol{\theta}}(\mathbf{w}), \mathcal{X}, \mathcal{Y}), \tag{5.6}$$

where ε_{LOO} denotes the LOO error (Eq. (4.45)) of the surrogate $\widehat{\mathcal{M}}(\mathbf{z}; \mathbf{w}, \mathcal{X}, \mathcal{Y})$ evaluated at $\{\mathcal{X}, \mathcal{Y}\}$ and $\widehat{\boldsymbol{\theta}}(\mathbf{w})$ denotes the optimal parameters of $\widehat{\mathcal{M}}$ for that particular $\mathbf{w}$ value. The term $\widehat{\boldsymbol{\theta}}(\mathbf{w})$ is calculated by solving the *inner loop optimisation* problem:

$$\widehat{\boldsymbol{\theta}}(\mathbf{w}) = \underset{\boldsymbol{\theta}\in\mathcal{D}_{\boldsymbol{\theta}}}{\arg\min}\ \varepsilon_{LOO}(\boldsymbol{\theta}; \mathbf{w}, \mathcal{X}, \mathcal{Y}). \tag{5.7}$$

The nested optimisation approach to DRSM comes with costs and benefits. On the one hand, each objective function evaluation of the outer-loop optimisation becomes increasingly costly w.r.t. the number of samples in the experimental design and the complexity of the surrogate model. On the other hand, the search space in each optimisation step can be significantly smaller, compared to the joint approach, due to the reduced number of optimisation variables. Moreover, this nested optimisation approach enables DRSM to be entirely non-intrusive. Off-the-shelf well-known surrogate modelling methods can be used to solve Eq. (5.7).

Albeit non-intrusive and having a relatively low dimension, the inner optimisation in Eq. (5.7) is in general the driving cost of DRSM. Indeed, calculating the parameters of a single high-resolution modern surrogate may require anywhere between a few seconds and several minutes. To reduce the related computational cost, it is often possible to solve *proxy surrogate* problems, *i.e.* using simplified surrogates that, while not being as accurate as their full counterparts, are computationally cheaper to calibrate. Once the nested optimisation problem above is solved, a high-accuracy surrogate $\widehat{\mathcal{M}}(\mathbf{z}; \widehat{\mathbf{w}}, \mathcal{X}, \mathcal{Y})$ is computed, on the resulting optimally compressed input space.

In case of kernel-based surrogates, such as Kriging, the low-accuracy proxies can be obtained by prematurely stopping the optimisation in the inner loop in Eq. (5.7) and/or by using isotropic kernels. For calculating the final high-accuracy Kriging surrogate, a high-computational budget optimisation

is performed instead, combined with the use of an anisotropic correlation family. The introduction of anisotropy is expected to improve the generalisation performance of the metamodel, as shown for instance in the study by Moustapha et al. (2018).

In case of polynomial chaos expansions, the proxy PCE surrogates assume uniformly distributed and independent input variables in $\mathbf{Z}$. The PCE coefficients are computed by solving Eq. (4.34) using the ordinary least squares method (Berveiller et al., 2006). To calculate the PCE coefficients of the final, high-accuracy, surrogate $\widehat{\mathcal{M}}(g(\mathbf{x};\widehat{\mathbf{w}}))$, the distributions of the input variables are fitted using kernel-smoothing, while retaining the independence assumption, motivated by the results in Torre et al. (2019). In addition, a sparse solution is obtained by solving the optimisation problem in Eq. (4.34) using least angle regression (Blatman and Sudret, 2011) instead of ordinary least squares.

5.4 APPLICATIONS USING KPCA, KRIGING AND PCE SURROGATES

5.4.1 *Introduction*

The performance of DRSM is evaluated on the following applications: (i) an artificial analytic function with 20 unstructured inputs and approximately known intrinsic dimension, (ii) a realistic electrical engineering model with 80 unstructured inputs and unknown intrinsic dimension and, (iii) a heat diffusion model with $16,000$ structured inputs and unknown intrinsic dimension.

For each benchmark application, DRSM is applied using KPCA for compression together with Kriging or polynomial chaos expansions for surrogate modelling. The surrogate performance is then compared, in terms of generalisation error, to the sequential application of unsupervised dimensionality reduction followed by surrogate modelling. To improve readability, various details regarding the implementation of the optimisation algorithms and the surrogate models calibration are omitted from the main text and given in Appendix C.1 instead. All the surrogate modelling tasks were carried out using the MATLAB-based uncertainty quantification software UQLAB (Marelli and Sudret, 2014, 2018; Lataniotis et al., 2018).

5.4.2 *Sobol' function*

The Sobol' function (also known as g-function) is commonly used as a benchmark in the context of uncertainty quantification. It reads:

$$Y = \prod_{i=1}^{M} \frac{|4X_i - 2| + c_i}{1 + c_i}, \tag{5.8}$$

where $\mathbf{X} = \{X_1, \ldots, X_M\}$ are independent random variables uniformly distributed in the interval $[0, 1]$ and $\mathbf{c} = \{c_1, \ldots, c_M\}^\mathsf{T}$ are non-negative constants. In this application, we chose $M = 20$ and the constants $\mathbf{c}$ given by Konakli and Sudret (2016a); Kersaudy et al. (2015):

$$\mathbf{c} = \{1, 2, 5, 10, 20, 50, 100, 500, 500, \ldots, 500\}^\mathsf{T}. \tag{5.9}$$

It is straightforward to see that the effect of each input variable X_i to the output Y is inversely proportional to the value of c_i. In other words, a small (resp. large) value of c_i results in a high (resp. low) contribution of X_i to the variance of Y. For the given values of the constants $\mathbf{c}$, one would expect that, roughly, the first 4 to 6 variables can provide a compressed representation of $\mathbf{X}$ with minimal information loss regarding the input-output relationship.

To showcase the performance of DRSM, an experimental design $\mathcal{X}$, consisting of 800 samples, is generated by Latin Hypercube sampling of the input distribution (McKay et al., 1979). Based on the samples in $\mathcal{X}$ and the corresponding model responses $\mathcal{Y}$, several combinations of KPCA, Kriging and PCE are tested within the DRSM framework. An additional set of 10^5 validation samples $\{\mathcal{X}_v, \mathcal{Y}_v\}$ is generated for evaluating the performance of the final surrogates.

The first analysis consists in comparing the generalisation performance as a function of the compressed input dimension m for Kriging and PCE models combined with KPCA with different kernels. Because of the availability of a validation set, the performance of the LOO error estimator in Eq. (4.45) is also assessed by comparing it with the true validation error in Eq. (4.41). Figures 5.2a and 5.2b show the LOO error estimator of the final surrogate model when using Kriging and PCE, respectively.

TABLE 5.2: Sobol' function: optimal DRSM configurations for Kriging- and PCE-based surrogate models

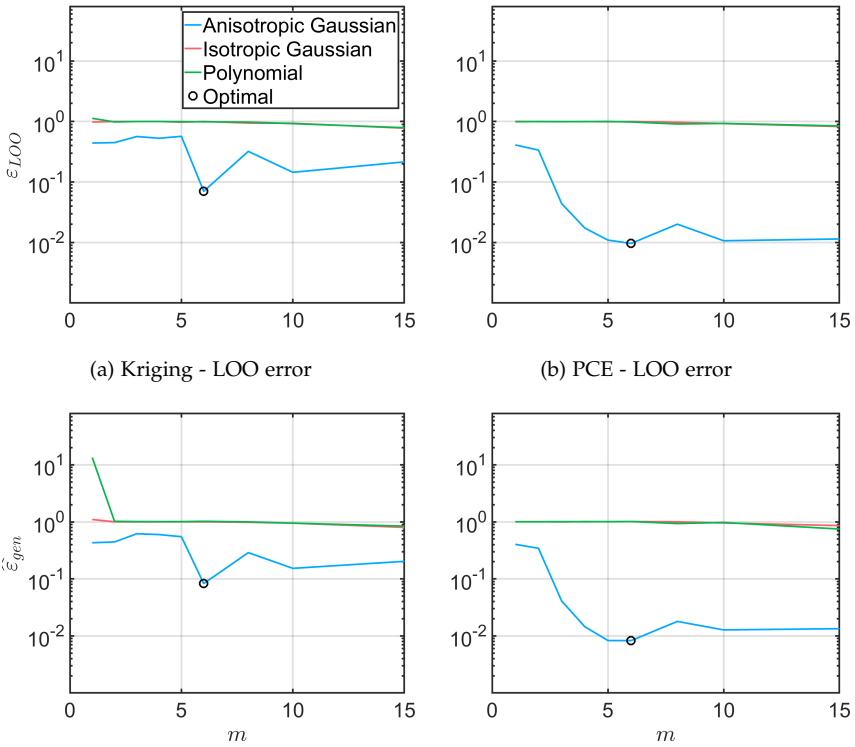
SM method	KPCA kernel	$\widehat{m}$	ε_{LOO}	$\widehat{\varepsilon}_{gen}$
Kriging	Anisotropic Gaussian	6	0.0704	0.0830
PCE	Anisotropic Gaussian	6	0.0096	0.0083

In each panel the different curves correspond to different KPCA kernels, namely polynomial kernel and isotropic (resp. anisotropic) Gaussian (see Table 3.1). Figures 5.2c and 5.2d show the corresponding validation error on the validation set for the same scenarios. At a first glance, it is clear that the top and bottom figures are remarkably similar, both in their trends and in absolute value. Therefore, it is concluded that on this application ϵ_{LOO} is a good measure of the generalisation error ϵ_{gen}. This is an important observation, because in the general case a validation set is not always available, while ϵ_{LOO} (or the related k-fold cross validation) can always be calculated. Moreover, the intrinsic dimension identified by all the best DR-SM combinations is equal to $\widehat{m} = 6$, which is a reasonable estimate based on the values of the constants c_i in Eq. (5.9). Indeed the total Sobol' indices of the 6 first parameters are equal to $\{0.6037, 0.2683, 0.0671, 0.0200, 0.0055, 0.0010\}$ and for the rest are in the order of 10^{-4}.

The DRSM algorithm identifies the anisotropic Gaussian kernel as the best KPCA kernel to be used in conjunction with both Kriging and PCE. However, the performance of PCE is significantly better in terms of generalisation error. The optimal parameters for each case (Kriging and PCE) are highlighted by a black dot in Figure 5.2, and their numerical values are reported in Table 5.2.

Subsequently, the performance of DRSM is compared against an unsupervised approach, in which dimensionality reduction is carried out first, before applying surrogate modelling. To facilitate a meaningful comparison between the various methods, the reduced dimension and the optimal KPCA kernel as determined by the first analysis (see Table 5.2) are used. The results are summarised in Figure 5.3, while the corresponding list of tested configurations for both DRSM and the sequential DR-SM is given in Table 5.3.

The experimental design consists of 800 samples. The performance of each method is evaluated in terms of the generalisation error of the final



(a) Kriging - LOO error

(b) PCE - LOO error

(c) Kriging - Validation error

(d) PCE - Validation error

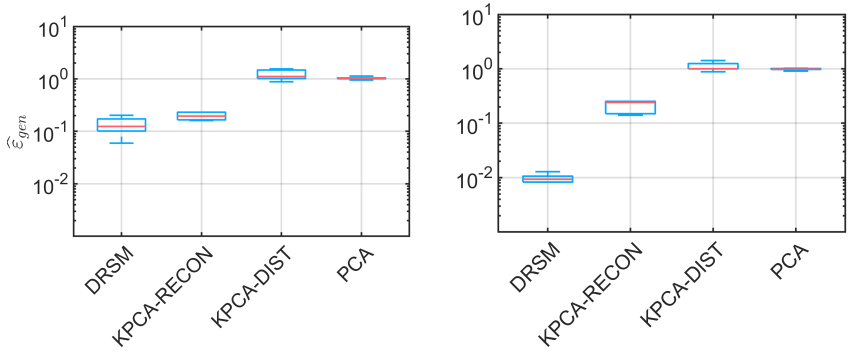
FIGURE 5.2: Error estimates of the DRSM surrogate as a function of the reduced space dimension. Kernel PCA is used with isotropic (resp.anisotropic) Gaussian as well as polynomial kernels.

surrogate $\widehat{\mathcal{M}}(\mathbf{z})$ evaluated on a validation set $\{\mathcal{X}_v, \mathcal{Y}_v = \mathcal{M}(\mathcal{X}_v)\}$ with 10^5 samples. To evaluate the robustness of the results, this process is repeated 10 times, each corresponding to a different set $\mathcal{X}$, drawn at random from $\mathbf{X}$ using the Latin Hypercube sampling method. On the left (resp. right) panel, a Kriging (resp. PCE) surrogate is calculated using one of the methods in Table 5.3. Each box plot in Figure 5.3 provides summary statistics of the generalisation error that was achieved by each configuration over the 10 repetitions. The central mark indicates the median, and the bottom and top edges of the box indicate the 25^{th} and 75^{th} percentiles, respectively.

TABLE 5.3: Different setups considered for evaluating the final surrogate model performance after using each of them for dimensionality reduction.

Dim. reduction	Parameter tuning objective	Abbreviation
Kernel PCA	ε_{LOO} of Kriging (KG) or PCE surrogate (Eq. (5.6))	DRSM
Kernel PCA	Reconstruction error (Eq. (3.26))	KPCA-RECON
Kernel PCA	Pairwise distance preservation (Eq. (3.22))	KPCA-DIST
PCA	-	PCA

The whiskers extend to the most extreme data points up to 1.5 times the inter-quartile range above or below the box edges.

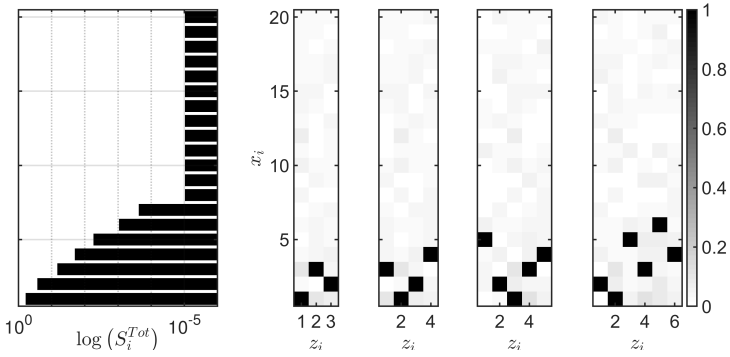


(a) Kriging (b) Polynomial chaos expansions

FIGURE 5.3: Sobol' function: estimates of the generalisation error.

The DRSM approach consistently shows superior performance compared to the unsupervised approaches. This performance improvement becomes more apparent in the case of PCE surrogate modelling, where the average validation error over the 10 repetitions is reduced by almost two orders of magnitude compared to the other methods.

Due to the analytical nature of the model under consideration, we further evaluate the DRSM-based input compression by means of how the most important input variables are mapped to the reduced space. We adopt the total Sobol' sensitivity indices as a rigorous measure of the importance of each input variable. Sobol' sensitivity analysis is a form of global sensitivity analysis based on decomposing the variance of the model output into contributions that can be directly attributed to inputs or sets of inputs (Sobol',



(a) Total Sobol' indices (b) $m = 3$ (c) $m = 4$ (d) $m = 5$ (e) $m = 6$

FIGURE 5.4: Sobol' function: visualisation of the sample-based Spearman correlation coefficient (absolute value) between the model inputs $\mathbf{X}$ and the reduced space inputs $\mathbf{Z}$.

1993). The total Sobol' sensitivity index of an input variable X_i, denoted by $S_i^{Tot} \in [0, 1]$, quantifies the total effect of X_i on the variance of Y. In this particular example, the total Sobol' indices can be analytically derived (Saltelli et al., 2000). Their values are shown for reference in Figure 5.4a.

It is clear from Eq. (5.8) and Eq. (5.9) that all 20 input variables contribute to the output variability, *i.e.* the intrinsic dimension of the problem is 20. However, the contribution of each input component quickly diminishes with larger values of c_i (see Figure 5.4a in which the values of the 20 total Sobol' indices are plotted, in logarithmic scale, as horizontal bars). Compressing the inputs in this problem is expected to lead to a mapping where those first few input components have the largest contribution.

In Figure 5.4 the features in the reduced space $\mathbf{Z}$ are compared against the original inputs $\mathbf{X}$. The rationale behind this heuristic analysis is simple: if the features obtained by DRSM are correctly identified, they should depend mostly on the same variables identified as important in the Sobol' analysis in Figure 5.4a. A simple measure of dependence between the reduced space components $\{z_i\,,\, i = 1, \ldots, m\}$ and the initial input space components $\{x_i\,,\, i = 1, \ldots, M\}$ is provided by the metric $|\rho\,(z_i, x_i)|$, where ρ denotes the Spearman correlation coefficient.

Figures 5.4b - 5.4e represent graphically the quantity $|\rho(z_i, x_i)|$ for the best surrogate identified in Table 5.2, namely a PCE coupled with KPCA using an anisotropic Gaussian kernel, evaluated on the validation set $\{\mathcal{X}_v, \mathcal{Y}_v\}$. Each figure corresponds to a different selection of reduced space dimension m. Figure 5.4 clearly shows that (i) each z_i correlates strongly with a specific x_i, (ii) the z_i's correlate with the m "most important" x_i's, and, (iii) the larger m value leads to the discovery of a new input z_i that correlates with the next "most important" component of $\mathbf{x}$.

5.4.3 *Electrical resistor network*

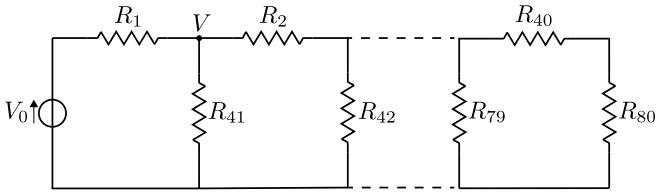
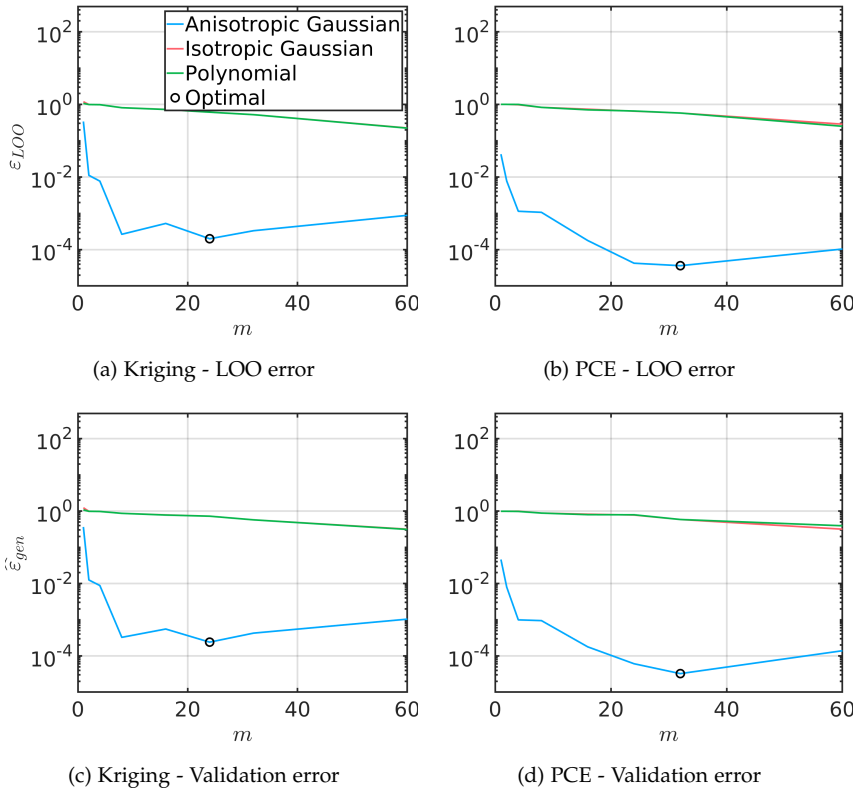


FIGURE 5.5: The resistor networks application example

The electrical resistor network in Figure 5.5 (Jakeman et al., 2015) is considered next. It contains 80 resistances of uncertain ohmage (model inputs), that are independent and uniformly distributed, and it is driven by a voltage source providing a known potential V_0. The output of interest is the voltage V at the node shown in Figure 5.5. A single set of $1,000$ experimental design samples and model responses is available, courtesy of J. Jakeman (Sandia National Laboratories).

As in the previous section, the goal of the first analysis is to determine the generalisation performance of the DRSM surrogate as a function of the reduced space dimension m when KPCA is combined with either Kriging or PCE. In addition, the accuracy of the LOO error in Eq. (4.45) is compared to the validation error in Eq. (4.42). The samples are randomly split into 500 pairs $\{\mathcal{X}, \mathcal{Y}\}$ used during the DRSM calibration and 500 pairs $\{\mathcal{X}_v, \mathcal{Y}_v\}$ used for validation.

Figures 5.6a and 5.6b show the LOO error estimator of the final surrogate model (Kriging or PCE), evaluated on $\{\mathcal{X}, \mathcal{Y}\}$, whereas Figures 5.6c and 5.6d show the validation error of the surrogate, evaluated on $\{\mathcal{X}_v, \mathcal{Y}_v\}$. In



(a) Kriging - LOO error

(b) PCE - LOO error

(c) Kriging - Validation error

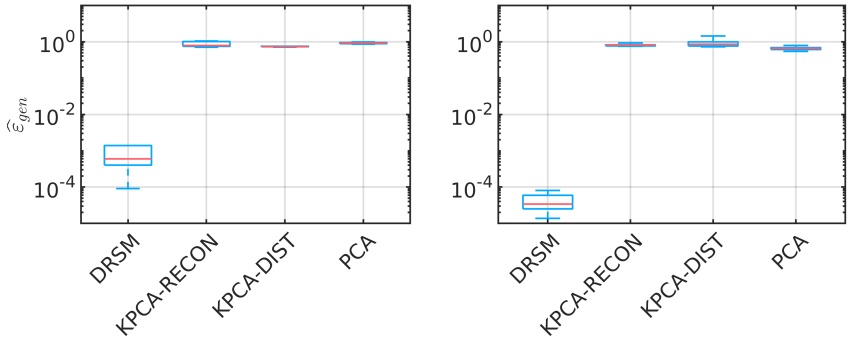
(d) PCE - Validation error

FIGURE 5.6: Electrical resistor networks: error estimates of the DRSM surrogate as a function of the reduced dimension. Kernel PCA is used with anisotropic (resp. isotropic) Gaussian as well as polynomial kernels.

each panel, each curve corresponds to a different KPCA kernel, namely anisotropic or isotropic Gaussian, and polynomial. Finally, the optimal configuration for each SM method is illustrated by a black dot. Similarly to the Sobol' function, the use of an anisotropic kernel in KPCA results in significantly reduced generalisation error. Indeed this is expected from a physical standpoint. The effect of the resistors on the voltage V will decay with distance (in terms of the number of preceding resistors) from V, which implies anisotropy in terms of the effect of each input variable to the output. As in the previous application example, the LOO error in Figures 5.6a and 5.6b provides a reliable proxy of the generalisation error in Figures 5.6c and

TABLE 5.4: Resistor networks: optimal DRSM configurations for Kriging and PCE surrogate models

SM method	KPCA kernel	$\widehat{m}$	ε_{LOO}	$\widehat{\varepsilon}_{gen}$
Kriging	Anisotropic Gaussian	24	2.000e-04	2.402e-04
PCE	Anisotropic Gaussian	32	3.621e-05	3.249e-05

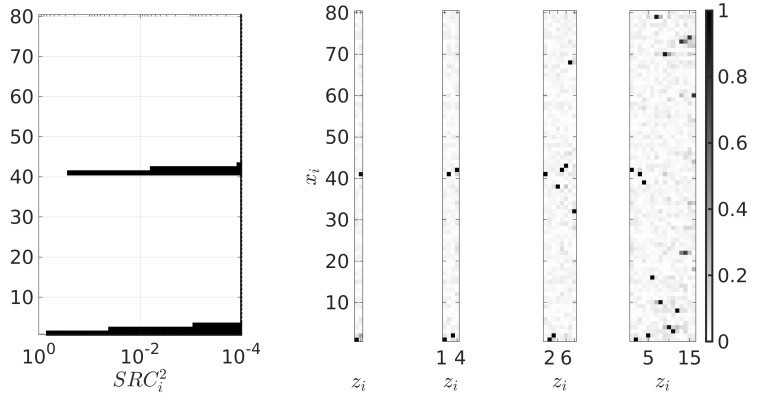


(a) Kriging (b) Polynomial chaos expansions

FIGURE 5.7: Electrical resistor networks: estimates of the generalisation error.

5.6d and the same optimal parameters are identified w.r.t. the two error measures. The optimal DRSM configuration for each surrogate model is given in Table 5.4.

Next, the performance of DRSM is compared to unsupervised approaches considering the setups in Table 5.3. The results of this comparative study are given in Figure 5.7 using box plots. They are obtained by the repeated random selection of 500 samples from the available 1,000, leading to 10 separate surrogate models for each case. The performance of each method is determined by means of the $\widehat{\varepsilon}_{gen}$ of the final surrogate $\widehat{\mathcal{M}}(\mathbf{z})$ evaluated on the validation set $\{\mathcal{X}_v, \mathcal{Y}_v = \mathcal{M}(\mathcal{X}_v)\}$, that corresponds to the remaining 500 samples of each split. Hence, each box-plot provides summary statistics of the validation error over the different splits. Each of the setups is tested both for Kriging (Figure 5.7a) and PCE surrogates (Figure 5.7b). In this application example, the DRSM-based surrogates outperform the others by several orders of magnitude in both cases (Kriging, PCE). This highlights the difference between the unsupervised and supervised compression:



(a) SRC coefficients (b) $m = 2$ (c) $m = 4$ (d) $m = 8$ (e) $m = 16$

FIGURE 5.8: Electrical resistor networks: visualisation of the sample-based Spearman correlation coefficient (absolute value) between the model inputs $\mathbf{X}$ and the reduced space inputs $\mathbf{Z}$.

compressing the input using only the information in $\mathcal{X}$ appears inefficient when followed by surrogate modelling.

Finally, we investigate how the reduced variables $\mathbf{Z}$ correlate with the input variables $\mathbf{X}$ in Figure 5.8. In contrast to the previous example, there is limited knowledge about the computational model, thus we proceed by approximating the importance of each input variable in the physical space using a sensitivity measure called standardised regression coefficients, denoted by SRC (initially introduced by Helton et al., 1985; used the UQLab implementation described in Marelli et al., 2019). The SRC are computed by first approximating $\mathcal{M}(\mathbf{X})$ by a linear model:

$$\mathcal{M}(\mathbf{X}) \approx \beta_0 + \sum_{i=1}^{M} \beta_i X_i \,, \tag{5.10}$$

where the coefficients $\boldsymbol{\beta} = \{\beta_0, \ldots, \beta_M\}$ are estimated by ordinary least-squares:

$$\widehat{\boldsymbol{\beta}} = \left(\mathcal{X}^\top \mathcal{X}\right)^{-1} \mathcal{X}^\top \mathcal{Y} \,. \tag{5.11}$$

The SRC indices are then defined as:

$$SRC_i = \frac{\widehat{\boldsymbol{\beta}}\widehat{\sigma}_{X_i}}{\widehat{\sigma}_Y}\ ,\ i = 1, \dots, M\,, \tag{5.12}$$

where $\widehat{\sigma}_i$ denotes the sample-based standard deviation of X_i and $\widehat{\sigma}_Y$ the sample-based standard deviation of Y. In Figure 5.8a we plot the squared SRC indices due to the property: $\sum_{i=1}^{M} SRC_i^2 = 1$ and because in this analysis we are not interested in the sign of each SRC_i but only in its magnitude [1]. Indeed variables X_1, X_2, X_{41} and X_{42} are expected to have the most significant contribution to the output variance because, as was previously pointed out, the effect of each resistor decays as the distance from V increases. Similarly to case of the Sobol' function, Figures 5.8b - 5.8e represent graphically the quantity $|\rho\,(z_i, x_i)\,|$, where ρ denotes the Spearman correlation coefficient, each panel corresponding to a reduced space of different dimension. In each case, we used the DRSM algorithm to calculate the parameters of the KPCA (anisotropic Gaussian) kernel which is coupled with a PCE surrogate, because this setup performed best in Table 5.4. Interestingly we observe that as the dimensionality of the reduced space, m, increases, the most important variables X_i are always captured by at least one Z_i, and the additional Z_i's either correlate with the next more important input variable or correlate with multiple X_i's when $m = 8$ or larger.

Further examination of those results in combination with the PCE prediction error results in Figures 5.6b and 5.6d, shows that for $m \leq 4$, a steep decline in the PCE generalisation error is observed because for each m value in that range, new features are discovered, each being important for the prediction of the model response because it corresponds to one of the X_i's with $i \in \{1, 2, 41, 42\}$. For $4 < m \leq 32$ the PCE still improves but in a reduced rate because the additional Z_i's indeed introduce further meaningful information about Y. Although the rest of the X_i's with $i \notin \{1, 2, 41, 42\}$ are less important in terms of their SRC indices, they still affect the value of the model response.

1 Note that this type of variance decomposition is meaningful only in cases where the input variables are uncorrelated, which is indeed the case in this application. When this assumption holds, the total variance of the model is given by (approximately, due to the linear approximation of computational model):

$$\widehat{\sigma}_Y^2 = \sum_{i=1}^{M} \beta_i^2 \sigma_{X_i}^2\,.$$

5.4.4 *2D diffusion*

This last application consists of a 2-dimensional stationary heat diffusion problem. The problem is defined in a square domain, $D = [-0.5, 0.5] \times [-0.5, 0.5]$, where the temperature field $T(\mathbf{v})$, $\mathbf{v} \in D$ is the solution of the elliptic partial differential equation:

$$-\nabla \cdot (d(\mathbf{v}) \nabla T(\mathbf{v})) = 500\, I_A(\mathbf{v}), \tag{5.13}$$

with boundary conditions $T = 0$ on the top boundary and $\nabla T \cdot \boldsymbol{n} = 0$ on the left, right and bottom boundaries, where $\boldsymbol{n}$ denotes the vector normal to the boundary. In Eq. (5.13), A corresponds to a square domain (see Figure 5.9) and I_A is the indicator function equal to 1 if $\mathbf{v} \in A$ and 0 otherwise. The diffusion coefficient $d(\mathbf{v})$ is a lognormal random field defined by:

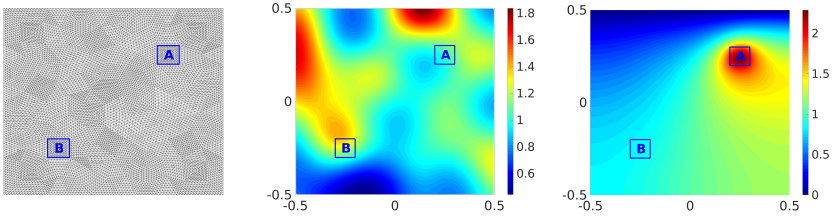
$$d(\mathbf{v}) = \exp\left(a_d + b_d\, g(\mathbf{v})\right), \tag{5.14}$$

where $g(\mathbf{v})$ is a Gaussian random field and the parameters a_d, b_d are such that the mean and standard deviation of d are $\mu_d = 1$ and $\sigma_d = 0.3$ respectively. The random field is characterised by a Gaussian correlation function $R(\mathbf{v}, \mathbf{v}') = \exp\left(-\|\mathbf{v} - \mathbf{v}'\|^2 / \ell^2\right)$, with $\ell = 0.2$. The output of interest is the average temperature in the square domain B within D (see Figure 5.9).

To solve Eq. (5.13), the Gaussian random field $g(\mathbf{v})$ is first discretised using the expansion optimal linear estimation (EOLE) method (Li and Der Kiureghian, 1993). Consider a grid in D with nodes $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\}$. By retaining the first p terms in the EOLE series, $g(\mathbf{v})$ is approximated by:

$$\widehat{g}(\mathbf{v}) = \sum_{i=1}^{p} \frac{\xi_i}{\sqrt{l^{(i)}}} \left(\boldsymbol{\phi}^{(i)}\right)^\top \boldsymbol{C}_{\mathbf{v}\boldsymbol{v}}(\mathbf{v}), \tag{5.15}$$

where $\{\xi_1, \ldots, \xi_p\}$ are independent standard normal random variables, $\boldsymbol{C}_{\mathbf{v}\boldsymbol{v}}$ is a vector with elements $C^{(k)}_{\mathbf{v}\boldsymbol{v}} = R(\mathbf{v}, \boldsymbol{v}_k)$ for $k = 1, \ldots, n$ and $\left\{\left(l^{(i)}, \boldsymbol{\phi}^{(i)}\right), i = 1, \ldots, n\right\}$ are the eigenvalues and eigenvectors of the correlation matrix $\boldsymbol{C}_{\boldsymbol{v}\boldsymbol{v}}$ with elements $C^{(i,j)}_{\boldsymbol{v}\boldsymbol{v}} = R(\boldsymbol{v}_i, \boldsymbol{v}_j)$ for $i, j = 1, \ldots, n$.



(a) Finite element mesh

(b) Single realisation of the input diffusion coefficient random field

(c) Corresponding temperature (delta) distribution

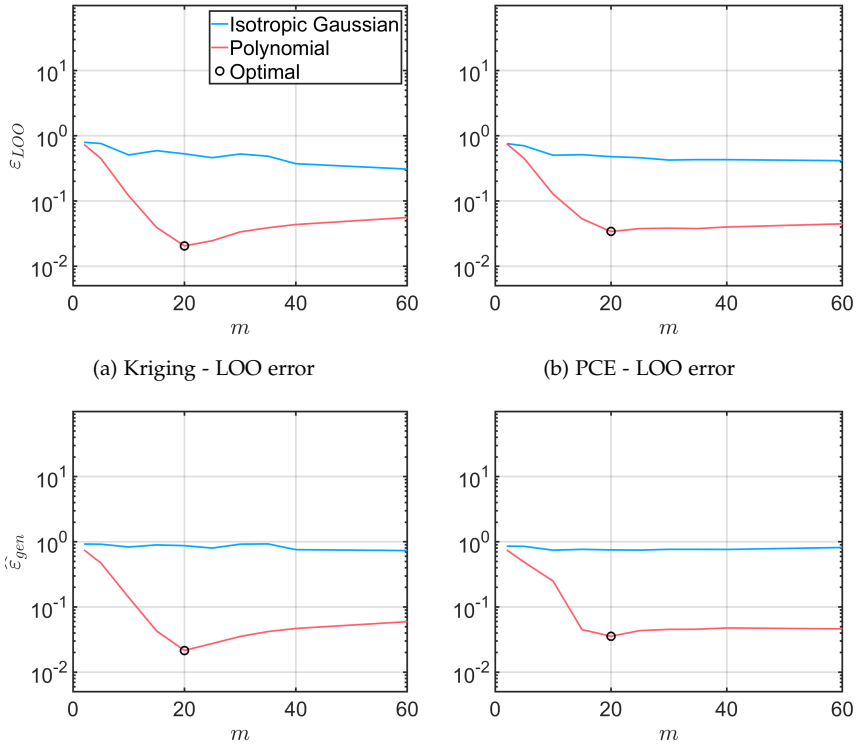
FIGURE 5.9: 2D heat diffusion problem: illustration of the model input and output.

TABLE 5.5: 2D diffusion: optimal DRSM configurations for Kriging- and PCE-based surrogate models

SM method	**KPCA kernel**	$\widehat{m}$	$\widehat{\mathbf{w}}$ (Table 3.1)			ε_{LOO}	$\widehat{\varepsilon}_{gen}$
			$\widehat{w}_1$	$\widehat{w}_2$	$\widehat{w}_3$		
Kriging	Polynomial	20	131.3681	112.0040	1	0.0205	0.0216
PCE	Polynomial	20	17.5225	15.1853	1	0.0340	0.0356

The underlying deterministic problem is solved with an in-house finite-element analysis code developed in MATLAB. The mesh shown in Figure 5.9a consists of 16,000 triangular T3 elements. Figure 5.9b shows a realisation of the diffusion coefficient random field which corresponds to the input of the model. The corresponding model output, shown in Figure 5.9c, is the mean temperature in the highlighted square region B. Each realisation of the diffusion coefficient random field is discretised over the mesh in Figure 5.9a. In the following analysis, the system is treated as a black-box, *i.e.* the discretised heat diffusion coefficient, obtained using Eq. (5.15) with $p = 20$ terms, is treated as a high-dimensional input ($M = 16,000$) and the average temperature in square B as the scalar model output. A single set of 500 experimental design samples and model responses is available. This example mimics a realistic scenario in which various maps of spatially varying parameters measured on a regular grid, are input to a computational model that analyses some performance of the system.

As in the previous application examples, the goal of the first analysis is to determine the optimal DRSM configuration in terms of the KPCA kernel and the reduced space dimension, as well as test the effectiveness



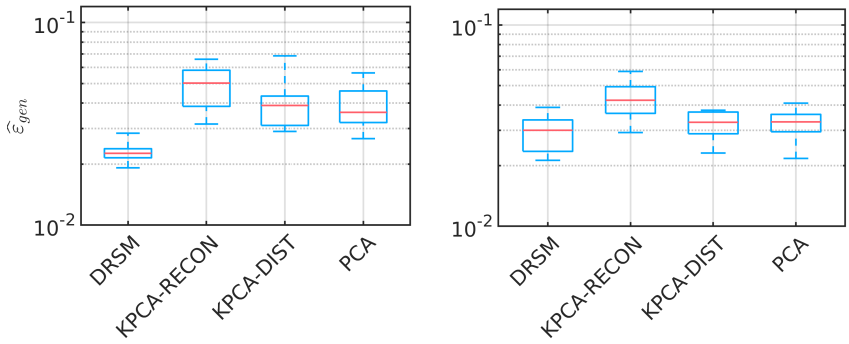
(a) Kriging - LOO error

(b) PCE - LOO error

(c) Kriging - Validation error

(d) PCE - Validation error

FIGURE 5.10: 2D heat diffusion problem: Error estimates of the DRSM surrogate as a function of the reduced space dimension. Kernel PCA is used with isotropic Gaussian and polynomial kernels.



(a) Kriging (b) Polynomial chaos expansions

FIGURE 5.11: 2D heat diffusion problem: estimates of the generalisation error.

of the LOO error as a proxy of the validation error. In this analysis, the available samples are randomly split into 300 pairs to be used during the DRSM optimisation and 200 pairs to be used for validation. The results are shown in Figure 5.10. Figures 5.10a and 5.10b show the LOO error estimator of the final Kriging (resp. PCE) surrogate, evaluated on $\{\mathcal{X}, \mathcal{Y}\}$, whereas Figures 5.10c and 5.10d show the validation error of the surrogate evaluated on $\{\mathcal{X}_v, \mathcal{Y}_v\}$. Each curve corresponds to a specific type of KPCA kernel, namely isotropic Gaussian and polynomial (see Table 3.1), and a specific surrogate, namely Kriging and PCE. We omitted the anisotropic Gaussian kernel for KPCA which is intractable due to the large input dimensionality.

A similar convergence behaviour is observed between Kriging- and PCE-based DRSM. The corresponding optimal parameter values are highlighted in Figure 5.10 and their numerical values are reported in Table 5.5. The linear polynomial kernel performs best in both cases and leads to the same reduced space dimension $\widehat{m} = 20$. This significantly low dimension can be interpreted by Eq. (5.15). The heat diffusion coefficient, although 16,000-dimensional, is a non-linear combination of p independent standard normal random variables. Moreover, the LOO and validation error curves show similar behaviour both in terms of their trend and their absolute value. Hence, the LOO error served as a reliable proxy of the validation error, as was observed in the previous application examples too.

In the subsequent analysis we compare the performance of the DRSM approach against other sequential approaches listed in Table 5.3. To test each setup, we repeat the calculation process 10 times. In each case the 500 available samples are split randomly into 300 samples for calculating the surrogate and 200 samples for validation. The optimal KPCA kernel that was determined by DRSM is used in all methods that involve KPCA. Also, for the sake of comparison, the same reduced space dimension $\hat{m} = 20$ is assumed for all methods.

The results of this comparative study are given in Figure 5.11 using box plots to provide summary statistics of the validation error over the different splits of the samples. In case of Kriging surrogate modelling, DRSM consistently provides superior results compared to the other methods. Notice that KPCA with linear kernel is equivalent to PCA on a scaled version of the experimental design with scaling factor $\sqrt{w_1}$ (see Appendix A.1 for more details). The Kriging surrogates, in contrast to the PCE ones, are affected by this scaling. This also explains the performance improvement compared to the case of PCA-based DR. In case of PCE surrogate modelling, the performance improvement gained by DRSM is marginal compared to PCA and KPCA with distance preservation- based tuning of $\mathbf{w}$.

Overall, DRSM consistently provides more accurate or at least comparable results compared to the other approaches. The main difference with a standard UQ setting in which the thermal conductivity is supposed to be sampled from a random field with *known properties*, is that the proposed DRSM methodology is purely data-driven, *i.e.* it would be applied identically in a case when the input maps are given without knowledge about the underlying random process.

Possible applications of this approach can be found in the field of hydrogeology. Nowadays, it is possible to sample complex permeability maps of the underground based on higher-order statistics (*i.e.* not only mean value and autocorrelation functions). The resulting maps, however, are obtained in a sample-based format, and serve as an input of a Darcy-type groundwater flow modelling code. The above mentioned machinery tested on the heat transfer problem is readily available for such applications.

5.5 CONCLUSION

Surrogate modelling is a key ingredient of modern uncertainty quantification. Due to the detrimental effects of high input dimensionality on most surrogate modelling techniques, the input space needs to be compressed to make such problems tractable. In this chapter a novel approach for effectively combining dimensionality reduction with surrogate modelling, called DRSM, was proposed. DRSM consists of three steps: (i) the DR and SM parameters are calculated by solving a nested optimisation problem, where only low-accuracy surrogates are considered to reduce the associated computational cost, (ii) the optimal configuration parameters, including the dimension of the reduced space, are estimated based on the surrogate model performance, and, (iii) a final high-accuracy surrogate is calculated using the optimal values of all the aforementioned parameters.

The performance of DRSM was compared on three different benchmark problems of varying complexity against the classical approach of tuning the dimensionality reduction and surrogate modelling parameters sequentially. DRSM consistently showed superior performance compared to the others in all the benchmark applications.

The novelty of the proposed methodology lies in its non-intrusive way of combining dimensionality reduction and surrogate modelling. This allows for the combination of various techniques without the need of tweaking the dedicated optimisation algorithms on which each of them capitalises. A practical implication of the non-intrusiveness of DRSM is that off-the-shelf surrogate modelling methods (or even software) with sophisticated calibration algorithms can be directly used within this framework.

The focus was given to data-driven scenarios where only a limited set of observations and model responses is available. We demonstrated that the leave-one-out cross-validation error of the surrogate models can serve as a reliable proxy for estimating the generalisation error in order to tune the DR parameters, but also to assess the overall accuracy of the resulting surrogate.

In application-driven scenarios, when the goal is to obtain a surrogate with optimal performance (regardless of its type) for a specific problem, the proposed approach can be extended by including the surrogate type itself as one of the DRSM optimisation parameters in Eq. (5.5). However,

special care is needed for the choice of the error metric used during the DRSM optimisation in this case, because the cross-validation-based error estimations by different surrogates may have different levels of bias (see *e.g.* Tibshirani and Tibshirani, 2009).

6

UNCERTAINTY PROPAGATION IN HIGH-DIMENSIONAL SYSTEMS

6.1 INTRODUCTION

Uncertainty propagation stands for the quantification of the uncertainty in the response of a system, modelled by a function $\mathcal{M} : \mathbb{R}^M \to \mathbb{R}$, due to the uncertainty in its input parameters. The uncertainties in the input parameters are modelled by a random input vector $\mathbf{X}$, with probability density function $f_{\mathbf{X}}$. Consequently, the response of the system $Y = \mathcal{M}(\mathbf{X})$ is also a random variable.

Uncertainty propagation ideally refers to the characterisation of the full probabilistic content of Y, *i.e.* the exact knowledge of its PDF f_Y. However, it is typically not possible to get the analytic formulation of f_Y in practice. Instead, various methods exist to get the approximate statistics of Y that can be broadly classified into three categories (Sudret, 2007): (i) *second moment methods* that deal with computing the mean and variance of the model response (or even higher order moments), (ii) *structural reliability methods* that try to determine the probability of extreme events by investigating the tails of the response PDF (Ditlevsen and Madsen, 1996; Melchers, 1999; Lemaire, 2009), and, (iii) methods that approximate the entire f_Y. Monte Carlo simulation is arguably the most well-known approach to achieve this. Alternative methods include *spectral methods*, that represent the complete response in an intrinsic way by using suitable tools of functional analysis (see polynomial chaos expansions in Section 4.3), or surrogate model-based methods.

The focus in this chapter is the approximation of the response PDF f_Y by Monte Carlo simulation. There are multiple challenges involved in applying even this fundamental technique in high-dimensional data-driven contexts. The accurate approximation of the output distribution may require several thousands or even millions of model evaluations, depending on

the accuracy required. However, in data-driven applications neither the computational model of the system, nor a probabilistic model of its inputs are available, but rather only a limited set of realisations of the input parameters (experimental design) and the corresponding model responses. This is a challenging scenario, especially when high input dimensionality is considered.

When the joint probability distribution of the input variables is unknown, as in data-driven applications, it is traditionally estimated by fitting complex probabilistic models. Although there are well-established approaches for fitting distributions to existing data, this task becomes non-trivial when the number of input variables is large (i.e. $\mathcal{O}\left(10^2 - 10^4\right)$). In addition, the number of unknown distribution parameters can be too large to be inferred from the limited amount of samples in the available data set (underdetermined problem). Even in the case that an adequate probabilistic input model is available, obtaining the corresponding model responses can pose a major challenge given that the true model is either not available or is computationally too expensive to be used to directly perform Monte Carlo simulation.

Efforts in high-dimensional uncertainty propagation are scarce, but this research topic becomes increasingly popular in recent years. When ordered inputs are considered, a somewhat richer literature is available. A typical workflow involves the use of global sensitivity analysis methods to select the subset of the input variables that mainly drive the output uncertainty (Iooss and Lemaître, 2015a). However, those methods typically suffer from the curse of dimensionality and it becomes challenging to apply them in $\mathcal{O}\left(10^2\right)$-dimensional spaces (see *e.g.* Sheikholeslami et al., 2019). For sparse problems, spectral techniques have been used for sensitivity and reliability analysis in $\mathcal{O}\left(10^2\right)$ dimensions using adaptive schemes to construct a sparse basis (Blatman and Sudret, 2011; Konakli et al., 2016; Deman et al., 2016; Zhang et al., 2017). For structured inputs, Soize and Ghanem (2017) propose a workflow for enabling polynomial chaos expansions-based uncertainty propagation in high dimensions using diffusion maps. Tripathy et al. (2016) propose a Gaussian process formulation with a built-in input compression scheme using active subspaces. The focus of that work was the surrogate model construction; standard Monte Carlo was then used for uncertainty propagation. Finally, within the context of deep-learning, Abdelaziz et al. (2015) propose approximation schemes that enable Monte Carlo-

based uncertainty propagation in deep neural networks while assuming Gaussian-distributed inputs, with focus on automated speech recognition applications.

In this chapter, we propose a novel method for high-dimensional uncertainty propagation by Monte Carlo simulation in data-driven contexts, capable to deal with both structured and unstructured inputs. The methodology capitalises on the DRSM algorithm introduced in Chapter 5. Using DRSM, the input space is compressed into a dimensionality that makes the inference of its probabilistic representation feasible. The general framework for data-driven uncertainty quantification proposed by Torre et al. (2018) is adopted, in order to deal with dependency structures in the reduced input space (potentially highly complex) using vine copula inference. This technique enables the generation of an arbitrary number of artificial input samples that accurately mimic the statistics of the experimental design. This approach is available even in the absence of a computational model, because DRSM also provides an optimal surrogate that operates in the compressed space. Having access to a surrogate model and due to its low computational cost, an arbitrarily large number of artificial output samples can be generated. The novelty of this methodology is twofold: (i) a novel input compression scheme is utilised, and, (ii) a novel technique for data-driven uncertainty quantification using vine copulas is adopted.

This chapter is structured as follows. In Section 6.2 an overview of the proposed methodology is given first, followed by a detailed description of each of its ingredients in Sections 6.2.1 to 6.2.4. Finally, in Section 6.3 the performance of this methodology is evaluated on several benchmark applications of increasing complexity.

6.2 OVERVIEW OF THE PROPOSED METHODOLOGY

Consider the experimental design $\mathcal{S} = \{\mathcal{X}, \mathcal{Y}\}$ that consists of the observations of the unknown, high dimensional input random vector $\mathcal{X} = \left\{\mathbf{x}^{(i)} \in \mathbf{X} \subseteq \mathbb{R}^M, i = 1, \dots, N\right\}$ and the corresponding, scalar, model responses $\mathcal{Y} = \left\{\mathcal{M}\left(\mathbf{x}^{(1)}\right), \dots, \mathcal{M}\left(\mathbf{x}^{(N)}\right)\right\}$. We consider the case that it is the only available information about the phenomenon under investigation. Moreover, the dimensionality of the input space is high, *i.e.* $M \sim 10^{2-4}$. The goal is to the estimate statistics of the response variable $Y = \mathcal{M}(\mathbf{X})$

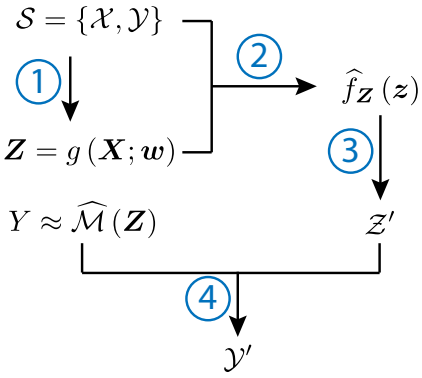


FIGURE 6.1: Overview of the proposed methodology for high-dimensional uncertainty propagation by reduced dimension resampling (RDR).

solely based on the available samples. Those statistics may include the full response PDF f_Y as well as summary statistics (moments and quantiles). We propose a methodology, called *reduced dimension resampling* (RDR) that consists in the four steps shown in Figure 6.1:

1. Application of the DRSM algorithm to the dataset $\mathcal{S}$: (i) the input space $\mathbf{X}$ is compressed in the reduced space $\mathbf{Z}$, and, (ii) a surrogate model that operates in the compressed input space is calibrated
2. The joint PDF $\widehat{f}_{\mathbf{Z}}$ of the reduced input vector $\mathbf{Z}$ is estimated based on the compressed experimental design
3. A large number of artificial samples $\mathcal{Z}'$ from $\sim \widehat{f}_{\mathbf{Z}}(\mathbf{z})$ is generated
4. The statistics of Y are inferred from the samples $\mathcal{Y}' = \widehat{\mathcal{M}}(\mathcal{Z}')$

Each step is discussed in more detail in Sections 6.2.1 to 6.2.4, respectively.

6.2.1 *Joint input compression and surrogate modelling*

Recall that the DRSM algorithm, introduced in Section 5.3, refers to the solution of the optimisation problem:

$$\left\{\widehat{\mathbf{w}}, \widehat{\boldsymbol{\theta}}\right\} = \underset{\mathbf{w}\in\mathcal{D}_{\mathbf{w}}, \boldsymbol{\theta}\in\mathcal{D}_{\boldsymbol{\theta}}}{\arg\min}\ \ell\left(\mathcal{M}(\cdot), \widehat{\mathcal{M}}\left(g(\cdot;\mathbf{w}), \boldsymbol{\theta}\right)\right), \tag{6.1}$$

where $\mathbf{w}$ denotes the parameters of the dimensionality reduction transformation $g : \mathbf{X} \to \mathbf{Z}$, $\boldsymbol{\theta}$ the parameters of the surrogate $\widehat{\mathcal{M}}$ and $\ell(\cdot)$ the loss function that quantifies the generalisation performance of the resulting surrogate.

After solving the optimisation problem in Eq. (6.1), both the input compression scheme $g\left(\cdot;\widehat{\mathbf{w}}\right)$ and the surrogate $\widehat{\mathcal{M}}\left(\cdot;\widehat{\boldsymbol{\theta}}\right)$ are available. For more details on the associated computational challenges and solution strategies, please see Section 5.3.

6.2.2 *Probabilistic input model inference in the compressed space*

After performing DRSM, a compressed experimental design is available, $\mathcal{Z} = g\left(\mathcal{X};\widehat{\mathbf{w}}\right)$ as well as the transform $g(\cdot)$. Provided that the dimension m of the reduced space is sufficiently small, it is likely tractable to estimate $f_{\mathbf{Z}}$ from a computational perspective. Although computationally tractable, such input model can be quite complex. Based on some elementary concepts from probability theory, notice that:

$$f_{\mathbf{Z}}(\mathbf{z}) = \frac{\partial}{\partial z_1}\cdots\frac{\partial}{\partial z_m}F_{\mathbf{Z}}(\mathbf{z}) \tag{6.2}$$

$$= \frac{\partial}{\partial z_1}\cdots\frac{\partial}{\partial z_m}\int_{\{\mathbf{x}\in\mathcal{D}_{\mathbf{X}}|g(\mathbf{x})\leq\mathbf{z}\}} f_{\mathbf{X}}(\mathbf{x})d\mathbf{z}\,. \tag{6.3}$$

Based on Eq. (6.3), even for a simple $f_{\mathbf{X}}$, the expression of $f_{\mathbf{Z}}$ can be significantly different (and complex) when a non-linear dimensionality reduction $g(\cdot)$ is used.

As discussed in Section 2.3.3, by virtue of Sklar's theorem any joint PDF can be expressed as a combination of its marginal distributions and a suitable copula that encapsulates the covariance structure:

$$f_{\mathbf{Z}}(\mathbf{z};\boldsymbol{\zeta}) = c\left(F_{Z_1}(z_1), \ldots, F_{Z_m}(z_m);\boldsymbol{\zeta}\right)\prod_{i=1}^{m} f_{Z_i}(z_i), \tag{6.4}$$

where $c(\cdot)$ denotes the copula density, F_{Z_i} (resp. f_{Z_i}) the marginal CDF (resp. PDF) of the i-th component of $\mathbf{Z}$ and $\boldsymbol{\zeta}$ the copula parameters. Hence, the approximation $\widehat{f_{\mathbf{Z}}}(\mathbf{z})$ of $f_{\mathbf{Z}}(\mathbf{z})$ can be obtained by solving two separate inference problems, one for the marginal distributions and one for the copula. Various techniques can be used to estimate the marginal CDF's F_{Z_i}. In this work, we use a non-parametric technique called kernel density estimation, described in more detail in Section 2.4.1.

Although a copula $c(\cdot)$ exists that satisfies Eq. (6.4), it is typically unknown in practice. In addition, when the dimension of $\mathbf{Z}$ (m) grows, it becomes increasingly difficult to describe pair-wise and higher-order dependencies (Torre et al., 2018). The work in Bedford et al. (2002); Torre et al. (2018) is adopted here to tackle this problem, that is based on specific copula constructions named *vine copulas*. Vine copulas describe the multivariate dependency in a random vector via products of simpler 2-copulas, *i.e.* copulas that describe the dependency between two conditional random variables. One challenge in this approach is that there is a significant number of possible pair-copula constructions. For example Aas et al. (2009) show that there are 240 different constructions for a five-dimensional density. Bedford et al. (2002) introduced the concept of regular vines that refer to an organised construction using a graphical model. In this work, we adopt one particular class of such a construction called *canonical* or *C-vines*, due to its flexibility and to the availability of data-driven inference strategies. A C-vine copula density is expressed as:

$$c(\boldsymbol{u}) = \prod_{j=1}^{M-1} \prod_{i=1}^{M-j} c_{j,j+i|\{1,\dots,j-1\}} \left(u_{j|\{1,\dots,j-1\}}, u_{j+i|\{1,\dots,j-1\}} \right), \tag{6.5}$$

where $u_i = F_{Z_i}(z_i)$ and $u_{i|\mathcal{A}} = F_{Z_i|Z_\mathcal{A}}(z_i|z_\mathcal{A})$ with $\mathcal{A}$ a general subset $\mathcal{A} \subset \{1, \dots, m\}$. For example, the C-vine density for a 4-dimensional random vector reads:

$$c(\boldsymbol{u}) = c_{1,2}\, c_{1,3}\, c_{1,4}\, c_{2,3|1}\, c_{2,4|1}\, c_{3,4|1,2}. \tag{6.6}$$

A shorthand notation is used in Eq. (6.6) with $c_{i,j|\mathcal{A}}$ corresponding to $c_{i,j|\mathcal{A}} \left(u_{i|\mathcal{A}}, u_{j|\mathcal{A}} \right)$. In this work, the pair copulas that are used to construct the C-vines and their associated parameters are listed in Table 6.1.

Name	$C(u, v; \zeta)$	Parameter range
Independence	uv	
Clayton	$(u^{-\zeta} + v^{-\zeta} - 1)^{-1/\zeta}$	$\zeta > 0$
Frank	$-\frac{1}{\zeta}\log\left(\frac{1 - e^{-\zeta} - (1 - e^{-\zeta u})(1 - e^{-\zeta v})}{1 - e^{-\zeta}}\right)$	$\zeta \in \mathbb{R}\backslash\{0\}$
Gaussian	$\Phi_{2;\zeta}\left(\Phi^{-1}(u), \Phi^{-1}(v)\right)$ $^{(a)}$	$\zeta \in (-1, 1)$
Gumbel	$\exp\left(-((-\log u)^{\zeta} + (-\log v)^{\zeta})^{1/\zeta}\right)$	$\zeta \in [1, +\inf)$
t-	$t_{2;\nu,\zeta}\left(t_{\nu}^{-1}(u), t_{\nu}^{-1}(v)\right)$ $^{(b)}$	$\nu > 1, \zeta \in (-1, 1)$

TABLE 6.1: Pair copulas considered in this work.

Fitting the C-vine entails three steps: (i) selecting the order of the u_i components in Eq. (6.5), (ii) selecting the parametric family for each pair copula, and, (iii) calculating the copula parameters $\boldsymbol{\zeta}$. Based on Torre et al. (2018), the algorithm by Aas et al. (2009) is used to solve step (i) first, which uses a heuristic approach to order Z_i in such a way that pairs $\left(Z_i, Z_j\right)$ with the strongest dependence are captured first. Then steps (ii) and (iii) are solved together by an iterative procedure. The pair copulas and their parameters are picked in such a way that the likelihood is maximised (see Section 2.4.2 for more details). Due to the complexity of the subject, a more detailed discussion about the strategies employed to identify the proper copula structure, as well as the specific numerical techniques to enable it are outside the scope of this work. For more details, the reader is referred to Aas et al. (2009); Torre et al. (2018). The numerical computations were performed within the UQLAB framework, through the upcoming Statistical Inference toolbox (Torre et al., 2019).

6.2.3 *Generation of artificial input samples*

The goal of this step is to generate a sufficiently large number of $\mathbf{Z}$ samples based on the joint distribution $\widehat{f}_{\mathbf{Z}}(\mathbf{z})$ inferred in the previous step. Various routines are available nowadays for generating random samples from the independent unit hypercube, *i.e.* $\boldsymbol{U} \sim \mathcal{U}\left([0, 1]^m\right)$. Such samples can be transformed in such a way that they follow other joint distributions using *isoprobabilistic transformations* (Lebrun and Dutfoy, 2009).

Consider first the special case of $\mathbf{Z}$ having independent components. Having access to the marginal distributions $F_{Z_1}, \ldots, F_{Z_m}$ (and their inverses) from the previous step, then the isoprobabilistic transform is carried out simply by:

$$\mathbf{Z} = \left\{ F_{Z_1}^{-1}(U_1), \ldots, F_{Z_m}^{-1}(U_m) \right\} . \tag{6.7}$$

A general approach for generating samples in the presence of dependence is the *inverse Rosenblatt transform* (Rosenblatt, 1952). Starting again from the uniformly distributed random vector $\boldsymbol{U}$, the components of $\mathbf{Z}$ can be calculated as follows:

$$Z_1 = U_1 \tag{6.8}$$
$$Z_2 = F_{2|1}^{-1}\left(U_2|Z_1\right) \tag{6.9}$$
$$Z_3 = F_{3|1,2}^{-1}\left(U_3|Z_1, Z_2\right) \tag{6.10}$$
$$\ldots = \ldots$$
$$Z_m = F_{m|1,,\ldots,m-1}^{-1}\left(U_m|Z_1, \ldots, Z_{m-1}\right) \tag{6.11}$$

where $F_{m|1,,\ldots,m-1}^{-1}\left(U_m|Z_1, \ldots, Z_{m-1}\right)$ is the cumulative distribution of the random variable $U_m|Z_1, \ldots, Z_{m-1}$. Although it may be challenging to calculate the conditional CDFs in the general case, Aas et al. (2009) provide an algorithm to compute the Rosenblatt transform and its inverse when the joint CDF of $\mathbf{Z}$ is described by a C-vine copula. In fact, C-vines provide an ideal framework for Rosenblatt transform because they allow the calculation of the conditional CDF's analytically.

6.2.4 *Estimation of the output statistics*

Starting from the artificial samples in the reduced space $\mathcal{Z}'$ that were generated in the previous step, and using the surrogate that was obtained after the application of the DRSM algorithm in the first step, the corresponding model responses are calculated as $\mathcal{Y}' = \widehat{\mathcal{M}}\left(\mathcal{Z}'; \widehat{\boldsymbol{\theta}}\right)$.

Using $\mathcal{Y}'$ we estimate the full response PDF, as well as its (empirical) mean, variance and quantiles. The underlying theory for such estimators is

standard in statistics textbooks, but a summary is given in Appendix A.3 for completeness.

6.3 APPLICATION EXAMPLES

To evaluate the performance of the proposed methodology for high dimensional uncertainty propagation problems, we revisit the benchmark applications from the previous chapter: (i) the Sobol' function, a commonly used benchmark function in the uncertainty quantification community, (ii) a realistic electrical engineering model with 80 unstructured inputs and, (iii) a heat diffusion model with 16,000 structured inputs.

In each case, the first step of applying DRSM (briefly recalled in Section 6.2.1) was already addressed in the previous chapter. The best performing surrogate (Kriging or polynomial chaos expansion) is used along with the optimal input compression parameters for kernel PCA. The subsequent steps are followed as described in Sections 6.2.2 to 6.2.4. We consider that the only available information about each system is a limited set $\mathcal{S}$ with $\mathcal{O}\left(10^2\right)$ samples. For benchmarking purposes, in each case an additional validation set of samples with size $\mathcal{O}\left(10^{5-7}\right)$ is available, denoted by $\{\mathcal{X}_v, \mathcal{Y}_v\}$. The validation set is considered sufficiently large to allow for the accurate estimation of the reference statistics of Y for validation purposes. To improve the readability of our findings, some detailed plots are omitted from the sections that follow and are shown in Appendix C.2 instead.

The implementation of the proposed approach took place in MATLAB and capitalised on various modules of the UQLAB software framework (Marelli and Sudret, 2014). In particular, the estimation of the probabilistic input model and the generation of additional artificial samples were based on the input and inference module (Lataniotis et al., 2019; Torre et al., 2019), while the surrogate models (polynomial chaos expansions and Kriging) were deployed using the corresponding UQLAB modules (Marelli and Sudret, 2018; Lataniotis et al., 2018).

6.3.1 *Sobol' function*

Recall the definition of Sobol' function from Eq. (5.8):

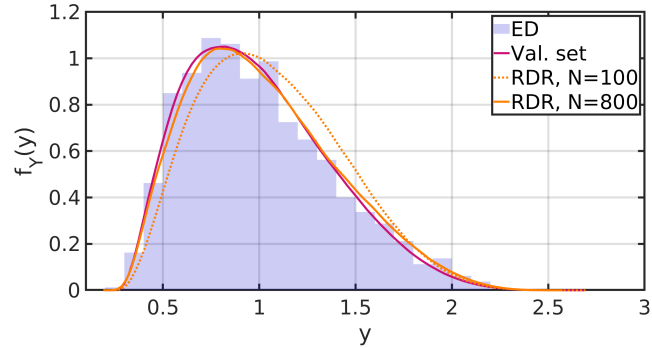
$$Y = \prod_{i=1}^{M} \frac{|4X_i - 2| + c_i}{1 + c_i}, \tag{6.12}$$

where the input random variable lies in the 20-dimensional unit hypercube, *i.e.* $\mathbf{X} \sim \mathcal{U}\left([0,1]^M\right)$ with $M = 20$ and the values of the constants $\boldsymbol{c} = \{c_1, \ldots, c_M\}^\top$ are given in Eq. (5.9).

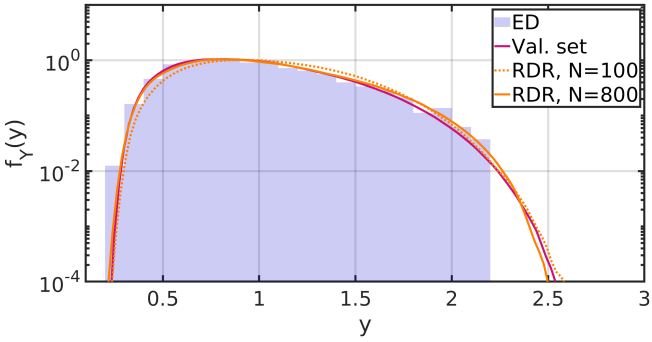
To showcase the performance of the proposed methodology, we estimate the statistics of Y using only a limited number of samples of $\{\mathcal{X}, \mathcal{Y}\}$ and without assuming any prior knowledge about the model, using the four step approach described in Section 6.2. Starting from the DRSM analysis performed in Section 5.4.2, we adopt the top performing setup from Table 5.2 for the subsequent calculations. This setup corresponds to compressing the input space from $M = 20$ to $m = 6$ dimensions using kernel PCA with an anisotropic Gaussian kernel and using a polynomial chaos expansions-based surrogate.

By capitalising on our findings in Figure 5.4, uniform distributions are used to model the marginals $f_{Z_i}, i = 1, \ldots, m$. As a reminder, it was observed that each $Z_i, i = 1, \ldots, m$ strongly correlates with a specific $X_i, i = 1, \ldots, 6$ which is uniformly distributed. A Gaussian copula is used to model the near-negligible dependencies in $\mathbf{Z}$, for simplicity. After fitting the input model, 10^7 artificial samples $\mathcal{Z}'$ are generated and fed to the PCE surrogate to obtain the corresponding responses $\widehat{\mathcal{Y}}'$. Those responses will be used to estimate various statistics of Y. The entire workflow is repeated for different sizes of the experimental design: $N \in \{100, 200, 400, 800\}$. For the sake of comparison, reference values of the output statistics are calculated using a validation set $\{\mathcal{X}_v, \mathcal{Y}_v\}$ with 10^7 samples that are generated using the true input and computational model.

The goal of the first analysis is to benchmark the proposed methodology in terms of the approximation quality of the output PDF, f_Y. The findings are summarised in Figure 6.2. Figures 6.2a and 6.2b show a graphical comparison between the reference PDF and the one estimated by the proposed approach using an experimental design of varied size, in linear and logarithmic scale, respectively. In all cases, the PDFs are estimated using the kernel density estimation method on the $\mathcal{Y}_v$ samples for the validation set and on $\mathcal{Y}'$ for the approximate ones. The kernel-density-based esti-



(a) Vertical axis in linear scale



(b) Vertical axis in logarithmic scale

FIGURE 6.2: Sobol function: comparison between the response PDF approximations using either true or artificial samples.

mates are obtained using a Gaussian kernel and the optimal bandwidth proposed by Silverman (2018). Finally, Figures 6.2a and 6.2b show the histogram that corresponds to the responses of the largest experimental design ($N = 800$).

The improvement in the PDF approximation by RDR when using $N = 800$ compared to using $N = 100$ samples, appears to be significant. Considering the former case, the lower tail of the distribution has been captured in a satisfactory degree in contrast to the upper tail region, where some deviation is observed. This discrepancy can be attributed to three factors: (i) the lossy compression from M to m dimensions, (ii) the approximate nature

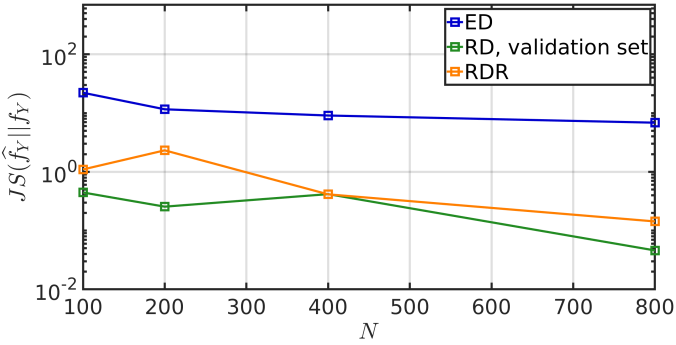


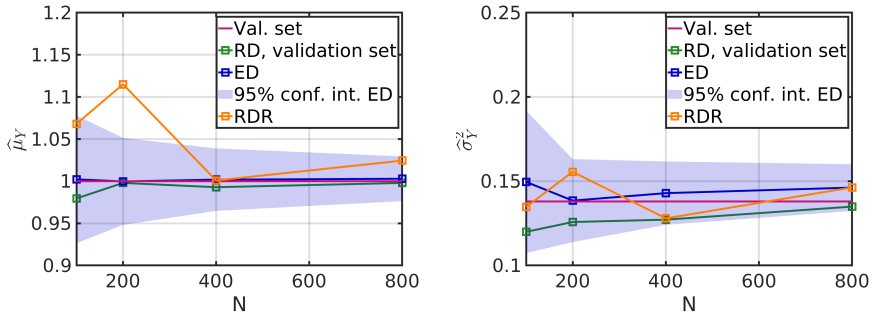
FIGURE 6.3: Sobol function: Jensen-Shannon divergence of f_Y estimates w.r.t. the validation samples.

of the surrogate model, and, (iii) the approximation error in the inference of $f_{\mathbf{Z}}$.

An additional quantitative evaluation of the f_Y estimates by RDR is shown in Figure 6.3. The approximate Jensen-Shannon (JS) divergence (Lin, 1991) is used to measure the difference between various PDF estimates and the reference one (see Appendix A.2 for more details). It is based on the Kullback-Leibler divergence with some notable differences, including that it always has a finite value and it is symmetric. The JS divergence values are non-trivial to interpret in an absolute sense but allow one to make comparisons between different estimates of the reference distribution.

The different curves in Figure 6.3 correspond to the JS divergence value that is obtained by estimating f_Y in one of the following ways: (i) by using directly the available samples $\mathcal{Y}$ with varying sample size (blue curve), (ii) by using the samples $\mathcal{Y}'$ that are generated using the proposed approach trained on the experimental design of varying size (orange curve), and, (iii) by using the samples $\widehat{\mathcal{M}}(g(\mathcal{X}_v;\widehat{\mathbf{w}});\widehat{\boldsymbol{\theta}})$, *i.e.* using the validation set instead of re-sampling (green curve).

At a first glance, the proposed approach improves the estimates of f_Y compared to simply using the experimental design. This improvement becomes more significant as the number of available samples increases. Notice that the distance of the green curve from the origin is due to the



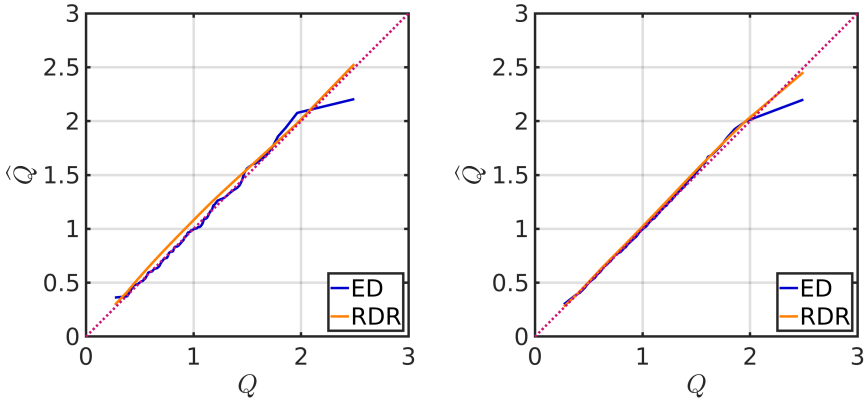
(a) Mean (b) Variance

FIGURE 6.4: Sobol' function: estimates of the output mean and variance.

input compression and the surrogate model error, *i.e.* due to the discrepancy between $\mathcal{M}(\mathcal{X}_v)$ and $\widehat{\mathcal{M}}(g(\mathcal{X}_v;\widehat{\mathbf{w}});\widehat{\boldsymbol{\theta}})$. Hence, the vertical distance between the orange and green curve is related to the probabilistic input model approximation error.

Subsequently, the proposed methodology is evaluated in terms of calculating various summary statistics of Y. In Figure 6.4 the mean (Figure 6.4a) and variance (Figure 6.4b) estimates are compared to their reference values inferred from the validation set, using an experimental design of varying size. In each panel, along with the reference value of the respective moment (red curve), the following moment estimates are plotted: (ii) the one obtained by $\widehat{\mathcal{M}}(g(\mathcal{X}_v;\widehat{\mathbf{w}});\widehat{\boldsymbol{\theta}})$ (green curve), (ii) the experimental-design-based estimate (blue curve), (iii) a shaded area indicating the 95% confidence interval on the experimental-design-based estimates based on Eq. (A.17), and (iv) and estimates obtained by $\mathcal{Y}'$ that is generated using the proposed methodology. Overall, using either $\mathcal{Y}$, or $\mathcal{Y}'$ gives comparable results, both in the mean and variance calculations. The resampling scheme seems to be typically introducing additional error, that may be minimal (variance, $N = 400$), with one exception (mean, $N = 400$) where it reduces the error introduced by the input compression and the surrogate.

The evaluation of the summary statistics is further expanded to the empirical quantiles in Figure 6.5. Using either $\mathcal{Y}$ or $\mathcal{Y}'$, the values of $\widehat{Q}(\alpha)$ are compared against the reference $Q(\alpha)$ for $\alpha \in [1/(N_v+1), N_v/(N_v+1)]$ sampled over a regular grid with 100 points, where N_v corresponds to



(a) $N = 100$ (b) $N = 800$

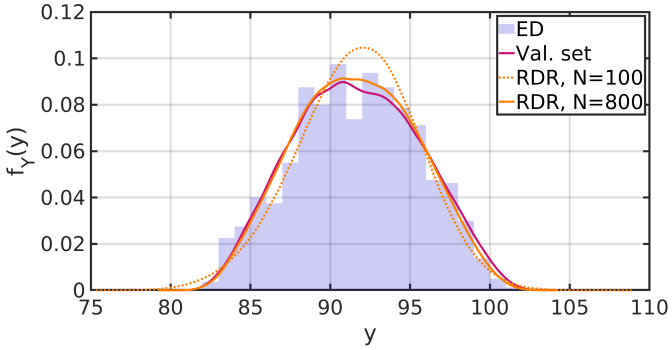
FIGURE 6.5: Sobol' function: quantile estimates comparison.

the number of samples in the validation set. The results corresponding to $N = 100$ (resp. $N = 800$) are shown in Figure 6.5a (resp. Figure 6.5b). The quantiles evaluated from $\mathcal{Y}$ appear to fluctuate around the reference values and show a significant deviation in the upper tail region. Both issues can be attributed to the limited number of samples. Using the artificial $\mathcal{Y}'$ samples results in a smoother curve with comparable error for the most part, but significant improvement is observed in both tails for both cases of $N = 100$ and $N = 800$. Overall, using $\mathcal{Y}'$ appears to enrich the limited samples in $\mathcal{Y}$ in such a way that the information about the statistics of Y is preserved or improved, especially in the extreme quantile regions.

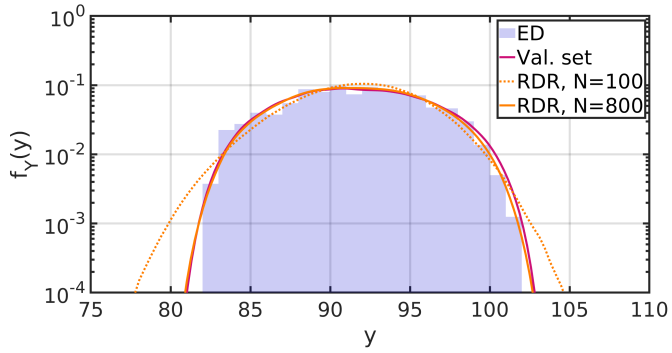
6.3.2 *Electrical resistor network*

The model of an electrical resistor network, described in Section 5.4.3, is considered next. The input $\mathbf{X}$ is an 80-dimensional random vector related to the ohmage of the network resistors and the output corresponds to the voltage at a specific location. In this example, fitting an 80-dimensional probabilistic input model is challenging, especially when dependencies need to be properly taken into account.

As in the previous section, the proposed RDR approach is showcased for varying size of the experimental design $N \in \{100, 200, 400, 800\}$. An



(a) Vertical axis in linear scale



(b) Vertical axis in logarithmic scale

FIGURE 6.6: Resistor networks: comparison between the output PDF approximations using either true or artificial samples.

available validation set $\{Xv, Yv\}$ with 10^5 samples is used to determine the reference values of the output statistics. The input compression and surrogate model parameters are adopted from the top performing setup given in Table 5.4, namely kernel-PCA- based dimensionality reduction with $m = 32$ and an anisotropic Gaussian kernel, and a polynomial-chaos-expansion- based surrogate.

The goal of the first analysis is to evaluate the output PDF estimate that is obtained by RDR. Figures 6.6a and 6.6b show a graphical comparison between the reference PDF from $\mathcal{Y}_v$ and the one estimated from the artificial output samples $\mathcal{Y}'$, using either a linear or logarithmic scale on the vertical

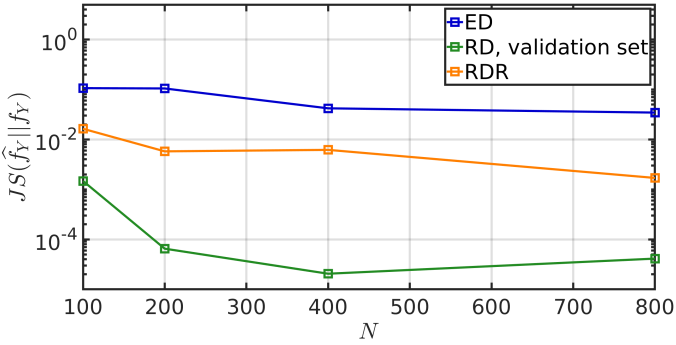
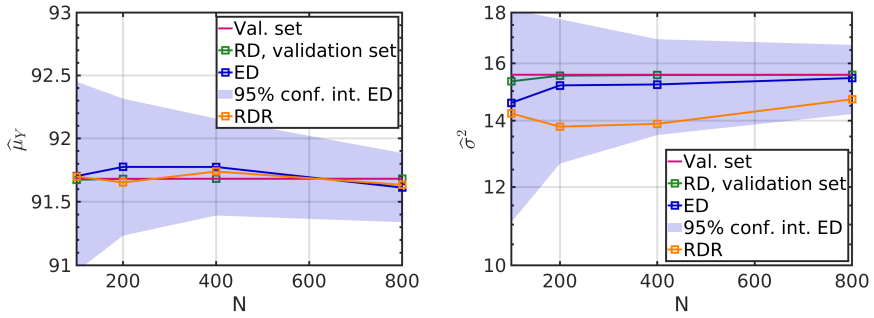


FIGURE 6.7: Resistor networks: Jensen-Shannon divergence of f_Y estimates w.r.t. the validation samples.

axis. Kernel density estimation with a Gaussian kernel is used in all cases to visualise the output PDF. A histogram of the given samples $\mathcal{Y}$ is also shown, considering the case $N = 800$. At a first glance, a significant improvement in $\widehat{f}_Y$ is observed when more samples are available, as expected. When $N = 800$, the lower tail of the distribution is captured accurately, but a non-negligible discrepancy between $\widehat{f}_Y$ and f_Y is observed for Y values approximately larger than 90 (volts).

In Figure 6.7 the overall fit of the output PDF is evaluated by means of the Jensen-Shannon divergence $JS(\widehat{f}_Y||f_Y)$ which is approximated using either the given samples $\mathcal{Y}$ (blue curve), the artificial samples $\mathcal{Y}'$ (orange curve), or $\widehat{\mathcal{M}}(g(\mathcal{X}_v;\widehat{\mathbf{w}});\widehat{\boldsymbol{\theta}})$, that use directly the validation set $\mathcal{X}_v$ bypassing the input model fitting and resampling steps (green curve). The overall fit using the proposed methodology improves by approximately one order of magnitude for all experimental design sizes compared to the one obtained by directly using the experimental design. Having perfect knowledge about the $f_{\mathbf{Z}}$ shows the potential of further improving $\widehat{f}_Y$ by about two additional orders of magnitude in terms of the JS divergence.

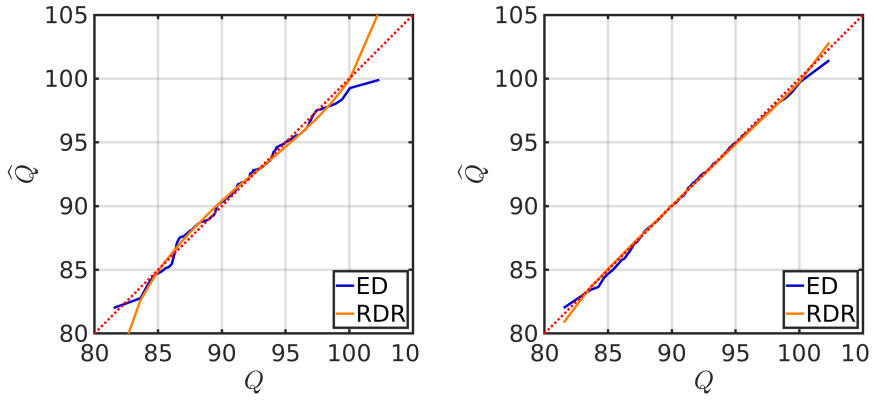
An in the previous application example, we proceed to evaluate the accuracy of various summary statistics estimates. Figure 6.8 shows the sample-based mean (Figure 6.8a) and variance (Figure 6.8b) convergence for varying sample size. The reference values (red curve) are obtained by processing $\mathcal{Y}_v$. Using $\widehat{\mathcal{M}}(g(\mathcal{X}_v;\widehat{\mathbf{w}});\widehat{\boldsymbol{\theta}})$ (green curve), achieves highly accurate results



(a) Mean (b) Variance

FIGURE 6.8: Resistor networks: estimates of the output mean and variance.

implying the error introduced by the input compression and surrogate modelling is relatively small. As expected, additional error is introduced when artificially generated samples are used instead (orange curve). However, there seems to be no benefit (or loss) using those samples instead of the given ones $\mathcal{Y}$ (blue curve) since both give comparable results.



(a) $N = 100$ (b) $N = 800$

FIGURE 6.9: Resistor networks: quantile estimates comparison.

The summary statistics evaluation continues in Figure 6.9 where we compare the empirical quantiles from $\mathcal{Y}$ and $\mathcal{Y}'$ to the reference ones from $\mathcal{Y}_v$. In particular, the values of $\widehat{Q}(\alpha)$ are compared against the reference

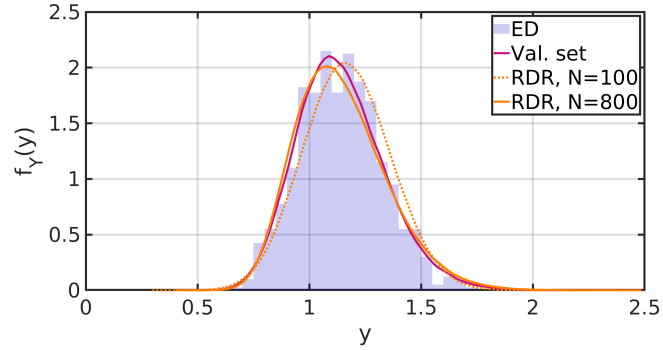
$Q(\alpha)$ for $\alpha \in [1/(N_v+1), N_v/(N_v+1)]$ sampled over a regular grid with 100 points. In Figure 6.9a (resp. Figure 6.9b) we use an experimental design of size $N = 100$ (resp. $N = 800$). Having access to more samples clearly improves our knowledge about the quantiles of Y especially in the tails, provided that the an adequate coverage of the input space has been achieved by the sampler. The proposed methodology achieves smoother but comparably accurate results when the sample size is too small. However, as the experimental design size increases, using $\mathcal{Y}'$ provides an overall improvement on the quantile estimates, especially in the extreme Y values compared to using $\mathcal{Y}$.

6.3.3 *2D diffusion*

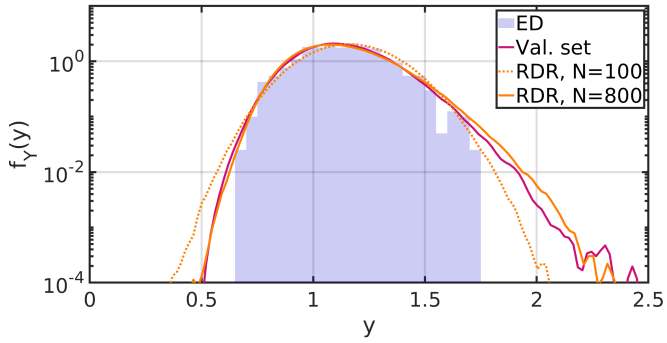
The last application consists of the 2-dimensional stationary heat diffusion model that is described in Section 5.4.4. From a black-box perspective, the input corresponds to a 16,000-dimensional random vector that contains the values of the heat diffusion coefficient over different positions on the plate, and the output corresponds to the mean temperature over a region of the plate. As a reminder, the high dimensional input is obtained by discretising the lognormal random field in Eq. (5.14) which describes the heat diffusion coefficient value at any point on the plate.

High-dimensional uncertainty propagation is performed using the proposed RDR methodology that results in the generation of 10^6 $\mathcal{Y}'$ samples that will be used to estimate various statistics of Y. The top performing setup from Table 5.5 is used, *i.e.* the compressed input space has dimension $m = 20$ and is calculated using kernel PCA with a polynomial kernel, while the surrogate model of choice is Kriging. This process is repeated for varying size of the $\{\mathcal{X}, \mathcal{Y}\}$ samples $N \in \{100, 200, 400, 800\}$. The reference values of the Y statistics are inferred from a validation set with size $N_v = 10^5$.

As in the previous examples, the goal of the first analysis is to evaluate the quality of the PDF estimator $\widehat{f}_Y$ that is obtained from $\mathcal{Y}'$ against the reference PDF from the validation set. The results are shown in Figure 6.10. In Figure 6.10a (resp. Figure 6.10b) a graphical comparison is made between the reference PDF from $\mathcal{Y}_v$ and its approximation by RDR for varying size of the experimental design using a linear (resp. logarithmic) scale on the vertical axis. In addition, the histogram of $\mathcal{Y}$ is shown from the sample of size $N = 800$, *i.e.* the largest one that was used. The PDF curves were



(a) Vertical axis in linear scale



(b) Vertical axis in logarithmic scale

FIGURE 6.10: 2D heat diffusion: comparison between the output PDF approximations using either true or artificial samples.

calculated using kernel density estimation with a Gaussian kernel on the respective sample sets. At a first glance, already from $N = 100$ (which is a significantly small size for this problem) the PDF approximation is close to the reference, with further improvement when $N = 800$.

This comparison is quantified in Figure 6.11 by means of the Jensen-Shannon divergence $JS(\widehat{f}_Y||f_Y)$ which is approximated using either the given samples $\mathcal{Y}$ (blue curve), the artificial samples $\mathcal{Y}'$ (orange curve), or $\widehat{\mathcal{M}}(g(\mathcal{X}_v;\widehat{\mathbf{w}});\widehat{\boldsymbol{\theta}})$ that use directly the validation set $\mathcal{X}_v$ and bypass the input model fitting and resampling steps (green curve). Overall, using $\mathcal{Y}'$ results in a JS divergence of approximately one order of magnitude smaller compared to using $\mathcal{Y}$.

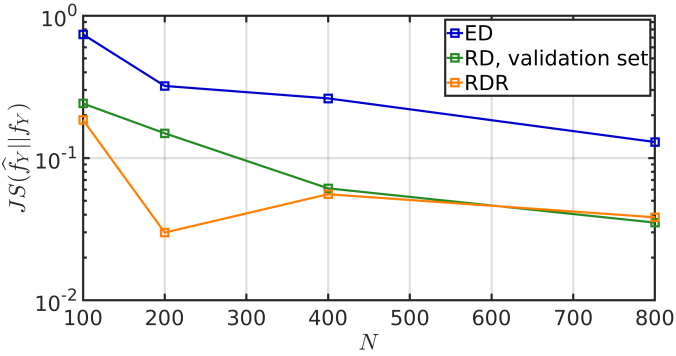
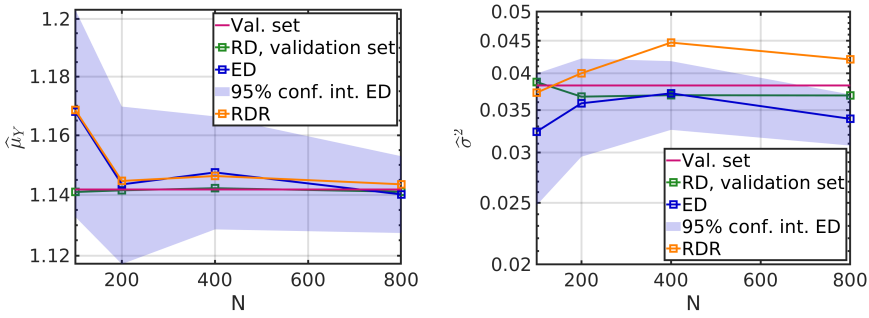


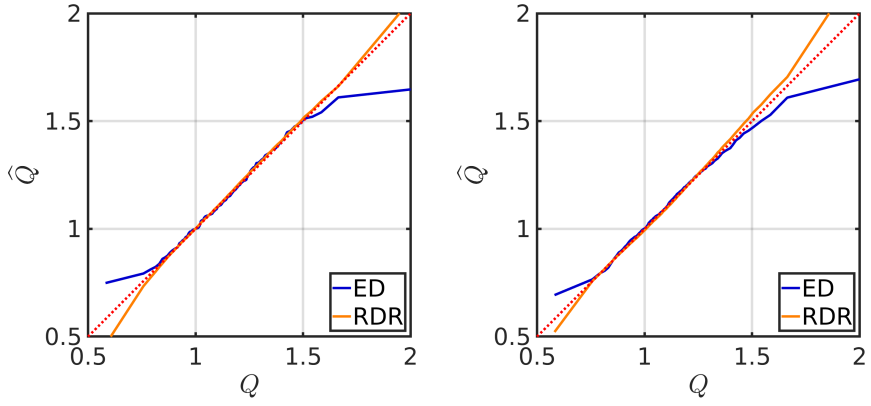
FIGURE 6.11: 2D heat diffusion: Jensen-Shannon divergence of f_Y estimates w.r.t. the validation samples.



(a) Mean (b) Variance

FIGURE 6.12: 2D heat diffusion: estimates of the output mean and variance.

Interestingly, the samples generated via the approximate probabilistic model $\widehat{f_{\mathbf{Z}}}$ result in an improvement on the Y PDF fit compared to $\widehat{\mathcal{M}}(g(\mathcal{X}_v;\widehat{\mathbf{w}});\widehat{\boldsymbol{\theta}})$. In other words, the probabilistic model fitting reduces the error instead of accumulating it on top of the input compression and surrogate modelling error. To understand why this happens, notice that the surrogate model is expected to perform well only within the boundaries of $\mathcal{Z} = g(\mathcal{X};\widehat{\mathbf{w}})$. Using the validation set, some samples of $\mathcal{Z}_v = g(\mathcal{X}_v;\widehat{\mathbf{w}})$ are likely to fall outside those boundaries, possibly leading to poor estimates of the output value. This also explains the fact that the discrepancy between the orange and green curve diminishes as N increases.



(a) $N = 200$ (b) $N = 800$

FIGURE 6.13: 2D heat diffusion: quantile estimates comparison.

In the second analysis, the proposed methodology is evaluated in terms of the quality of its mean and variance estimates. In Figure 6.4 the mean (Figure 6.4a) and variance (Figure 6.4b) estimates are compared to their reference values inferred from the validation set, using an experimental design of varying size. In each panel, along with the reference value of the respective moment (red curve), the following moment estimates are plotted: (ii) the one obtained by $\widehat{\mathcal{M}}(g(\mathcal{X}_v; \widehat{\mathbf{w}}); \widehat{\boldsymbol{\theta}})$ (green curve), (ii) the experimental-design-based estimate (blue curve), (iii) a shaded area indicating the 95% confidence interval on the experimental-design-based estimates based on Eq. (A.17), and (iv) the estimates obtained by $\mathcal{Y}'$ that is generated using the proposed methodology. Regarding the mean, using $\mathcal{Y}'$ results to almost identical values to the ones obtained by $\mathcal{Y}$. The same cannot be said for the variance estimates. Resampling introduces a larger error in the latter case making the sample-based variance of $\mathcal{Y}'$ to deviate more significantly from the reference value.

Finally, we evaluate the empirical quantiles that are obtained using the proposed approach in Figure 6.13. Using either $\mathcal{Y}$ or $\mathcal{Y}'$, the values of $\widehat{Q}(\alpha)$ are compared against the reference $Q(\alpha)$ for $\alpha \in [1/(N_v+1), N_v/(N_v+1)]$ sampled over a regular grid with 100 points. The results corresponding to $N = 200$ (resp. $N = 800$) are shown in Figure 6.13a (resp. Figure 6.13b). Overall, $\mathcal{Y}'$ provides an improvement on the accuracy of the empirical quan-

tiles, sometimes more significant (see the entire curve when $N = 200$ and the lower tail when $N = 800$) than others (higher tail when $N = 800$).

6.4 CONCLUSION

Data-driven uncertainty propagation is an emerging research field, especially when a high-dimensional input space is considered. On the one hand, "data-driven" refers to the limited knowledge about: (i) the equations that drive an engineering system, and, (ii) the uncertainty of its input parameters. One often only has access to samples of the input parameters and the corresponding model responses. On the other hand, the large number of input parameters renders various state-of-the-art techniques in modern uncertainty quantification non-applicable. This applies to the calculation of surrogate and of probabilistic input models, both suffering from the curse of dimensionality. In this chapter, we propose a workflow to overcome those challenges by capitalising on the optimal input compression and surrogate modelling scheme, DRSM, that was introduced in Chapter 5.

It was shown how one of the fundamental uncertainty propagation techniques, Monte Carlo simulation, can be enabled on such challenging problems. Having access to a compressed representation of the input space of manageable size, we used well-established techniques for resampling from that input space. Either due to the nature of the problem or the non-linearity of dimensionality reduction, the joint probability distribution of the reduced input space may be quite complex. To that end, we adopted the copula formalism to model the dependency structure of the inputs, and in particular canonical vine copulas, because they allow for data-driven, computationally efficient and scalable inference of their parameters.

The performance of the proposed methodology was evaluated on three different benchmark problems of varying complexity. We demonstrated that our knowledge about the output uncertainty improves when this method is used compared to the one that is extracted from the experimental design. This applies to the estimation of the overall shape of the output PDF as well as its empirical quantiles. Furthermore, this improvement was observed consistently in all the application examples and it was especially notable in the extreme value regions.

It is noteworthy to mention that this workflow can be modified to accommodate other uncertainty propagation techniques. Having access to a compressed input space of relatively small size, its probabilistic representation and a surrogate model, the problem becomes well-defined within the the classical UQ framework.

7

ENGINEERING APPLICATION: STRUCTURAL HEALTH MONITORING OF A WIND TURBINE SYSTEM

7.1 INTRODUCTION

Dealing with high-dimensional models is often the case nowadays in the engineering practice. It is particularly common in structural health monitoring (SHM), where long series of sensor readings need to be processed to asses the health state of a complex system in different time instances (present or future). In this chapter, we are considering the problem of SHM of wind turbines.

Sustainable means of energy production are becoming increasingly relevant to offset the environmental impact of traditional means of energy production. For example, the European Union (EU) has published a renewable energy roadmap, according to which a mandatory target is set of 34% renewable energy share by 2020, with wind energy supplying up to 14% and the rest 20% provided by other sources such as hydro, biomass and solar energy (Wilkes et al., 2011). Based on the response of the wind energy industry, four years later the Energy Roadmap 2050 predicted that wind energy could supply between 31.6% and 48.7% of Europe's electricity (European, commission, 2011). Wind turbines have been significantly refined in terms of materials and the design of various sub-systems, but their lifecycle management is still at its infancy based on the short life span of various components and the lack of efficient operation and maintenance schemes. The cost of the latter may in fact account for up to 25-30% of the total cost per KWh over the lifetime of a wind turbine, or 75-90% of the investment costs (Vachon, 2002).

Devising operation and maintenance strategies for wind turbine facilities is in fact highly challenging. First, a wind turbine is subjected to diverse operational loads. In addition, there is an inherent complexity to the system itself, because it consists of several structural and mechanical components.

These components are exposed to intensive load combinations such as wind gusts, turbulence, sea waves and corrosive environments. Such diverse loading may lead to various failure modes, *e.g.* due to structural damage, collapse from turbine overspeed, resonances, or severe localised vibrations. Consequently, such costly structures are designed to perform for a relatively short lifespan of approximately $20 - 25$ years, as a compromise between reliability and construction costs.

Applying modern SHM methodologies can provide significant benefits in terms of safety, down-time and maintenance costs of wind turbines. From this broad topic, the focus in this chapter is given on the development of novel algorithms, primarily for data-driven estimation of the fatigue accumulation and peak load in various system components. The importance of such algorithms is paramount, since they can enable the prediction of the residual fatigue lifetime of the components and their reliability. Considering an operational wind turbine, having access to such information, would allow the operators to minimise the down-time of the system and reduce the frequency of sudden breakdowns, as well as the associated maintenance and logistic costs (Agbayani, 2010; Grasse et al., 2011).

In an effort to mimic such a scenario, we first generate a set of realistic inflow wind measurements. For each of those we compute the corresponding indicators related the fatigue accumulation and peak loads on selected components of a wind turbine using specialised simulation software. Having access to a limited amount of such data, we showcase how the tools introduced in the previous chapters can be deployed to tackle two fundamental challenges: (i) estimating the fatigue accumulation and peak loads when the system is subjected to a new wind climate, and, (ii) quantifying the uncertainty of such loads, which is the key-ingredient to estimate the reliability of those components, *i.e.* how likely they are to fail in various future time instances.

The chapter is structured as follows. In Section 7.2 we describe the system under consideration and the methods and tools that we used to generate a realistic dataset. The first challenge of devising fatigue accumulation and peak load estimators is addressed in Section 7.3. Due to the high dimensionality of the underlying problem, we deploy several compression schemes that are tailored for this application and use the DRSM algorithm introduced in Chapter 5 to calibrate the associated parameters. The second challenge of quantifying the uncertainty of the quantities of interest is

addressed in Section 7.4, by applying the reduced dimension resampling technique that is presented in Chapter 6.

7.2 MODEL DESCRIPTION

7.2.1 *Wind climate*

An important aspect in wind turbine design is the wind conditions on the particular site the system will be deployed onto. Due to the variability of those conditions among different sites, so-called *design classes* have been introduced to enable manufacturers to design generic wind turbines without knowing the exact conditions of each particular site. They are specified by the IEC 61400-1 standard. A wind turbine manufactured according to a design class is considered viable to be installed on a site where the wind conditions do not exceed the ones specified by this class.

Two basic parameters are involved in the specification of each design class: (i) the reference wind speed V_{ref}, and, (ii) the turbulence intensity at wind speed $15\,m/s$ denoted by I_{ref}. According to the IEC 61400-1 standard, the mean wind speed at the hub height, V_{hub} is assumed to follow a Rayleigh distribution with CDF:

$$F(V_{hub}) = 1 - \exp\left[-\pi\left(\frac{V_{hub}}{2V_{ave}}\right)^2\right], \tag{7.1}$$

with V_{ave} chosen as:

$$V_{ave} = 0.2V_{ref}. \tag{7.2}$$

The values of each parameter for a specific class are listed in Table 7.1. Based on the table, a design class *e.g.* III_A corresponds to $V_{ref} = 37.5\,m/s$ and $I_{ref} = 16\%$. The reported design classes in Table 7.1 reflect common conditions excluding offshore sites, as well as extreme wind climate that is encountered during tropical storms such as hurricanes, cyclones and typhoons.

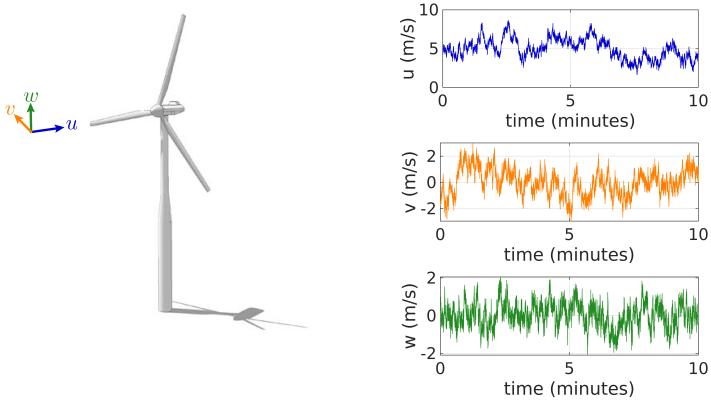
In this work we consider wind climates of class II_B as it is standard for wind turbine designs to be deployed in mainland and coastal Europe.

The TurbSim software is used to simulate the wind climate of the selected II_B design class (Jonkman, 2009). TurbSim is a stochastic, full-

TABLE 7.1: Parameter values for different wind turbine design classes.

Class	I	II	III
$V_{ref}\,(m/s)$	50.0	42.5	37.5
A $I_{ref}\,(\%)$		16	
B $I_{ref}\,(\%)$		14	
C $I_{ref}\,(\%)$		12	

field, turbulent-wind simulator. It uses a statistical model (as opposed to a physics-based model) to numerically simulate time series of three-component wind-speed vectors at points in a two-dimensional vertical rectangular grid that is fixed in space.



(a) The three components of the wind speed vector

(b) A realisation of each wind component

FIGURE 7.1: The wind velocity vector that is generated using TurbSim.

For simplicity, we consider a single measurement point of the wind velocity vector, located at the rotor hub of the wind turbine (see Figure 7.1a). Hence, the generated wind speed realisations consist of three time series, each referring to one component of the wind velocity vector. The simulation time of each realisation is 10 minutes and the sampling frequency is 20 Hz. An example of such a realisation is shown in Figure 7.1b.

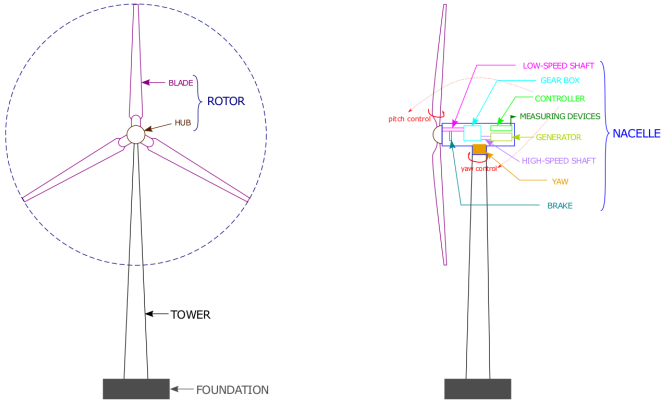


FIGURE 7.2: A schematic representation of an upwind horizontal axis wind turbine.

7.2.2 *Wind turbine model*

A specific design family of wind turbines is considered, namely upwind horizontal axis wind turbines (HAWT), depicted in Figure 7.2. The upwind HAWT structure mainly consists of three components: (i) the rotor with the three attached blades, which is mounted to the hub via the pitch bearing system that allows the blades to turn; (ii) the nacelle, which is attached to the tower and performs the conversion of mechanical to electrical energy; (iii) the yaw bearing system, that allows to adjust the rotor direction depending on the wind flow vector.

During the operation of a wind turbine, the rotor is driven by the wind energy. The low-speed shaft rotating with the rotor connects to the gear box, which increases the rotation speed to the high-speed shaft. The generator then produces electric energy from the kinematic power passed by the high-speed shaft. In case of emergency, the brake can stop the rotor. The nacelle contains a controller which determines how the turbine responds to different wind conditions via the generator-torque, the blade-pitch and the nacelle-yaw control, based on the wind climate measured by the attached sensors. Considering the efficiency of energy production and the safety requirements on energy production structures, a turbine starts to operate at the so-called cut-in wind speed u_{in} to avoid unstable power production,

reaches its rated power production at the rated speed u_{rat} and stops working at the cut-out speed u_{out}.

In this work, we consider a baseline turbine that is commonly used due to its resemblance with existing prototypes. It is known as 5 MW baseline turbine with the characteristics specified in Table 7.2 (Jonkman et al., 2009).

TABLE 7.2: Specifications of the 5 MW baseline wind turbine.

Rating	5 MW
Rotor orientation, configuration	Upwind, three blades
Control	Variable speed, collective pitch
Drivetrain	High speed, multiple stage gearbox
Rotor diameter	126 m
Hub diameter	3 m
Hub height	90 m
u_{in}	3 m/s
u_{rat}	11.4 m/s
u_{out}	25 m/s
Cut-in, rated rotor speed	6.9 rpm, 12.1 rpm
Rated tip speed	80 m/s
Overhang, Shaft tilt, precone	5 m, 5^o, 2.5^o
Rotor mass	110,000 kg
Nacelle mass	240,000 kg
Tower mass	347,000 kg

We use the OpenFAST simulation software (Jonkman, 2013) to calculate the loads on various points of interest of the wind turbine, given the wind velocity vector time-series generated with TurbSim. OpenFAST performs an aero-servo-elastic simulation that entails several computational steps. First, the aerodynamics sub-model converts the external wind fields to loads on the structure which is represented by a finite element model. The dynamic properties of the structure are determined based on the material properties and the actions of the control system that depend on the wind conditions at each time instance. The system response is then calculated through structural and multi-body dynamics. Moreover, each response interacts with the wind field and is taken into account for the consecutive simulation step.

7.2.3 *Fatigue and damage equivalent loads*

Wind turbines are dynamic systems exposed to aerodynamic loading and quasi-periodic excitation from the rotor. Structural components typically face $\mathcal{O}\left(10^{8-9}\right)$ load cycles over their lifetime, which roughly corresponds to 20-25 years. Hence, they are particularly susceptible to fatigue damage.

A wind turbine component is typically subject to several simultaneous loads. However, only one of them is considered in the subsequent fatigue analysis for simplicity. Consider a periodic load, *e.g.* moment, denoted by $M(t)$ and t is the time instance. First assume that the load is of the form $M(t) = A\sin(\omega t)$ with a fixed amplitude A and fixed cycle period $2\pi/\omega$. One way to model the lifetime of the component when such a periodic load is applied is based on the *S-N* or *Wöhler curve* (Suresh, 1998), which assumes the following relationship:

$$\log N(A) = \log K - m\log(A) \Leftrightarrow N = \frac{K}{A^m} \ , \tag{7.3}$$

where $N(A)$ denotes the number of cycles until failure when a sinusoidal load of amplitude A is applied, and the constants K and m depend on the component's material. In practice, the periodic load $M(t)$ is more complex in terms of its frequency spectrum rather than a single sinusoid. However, the material model defined under the sinusoidal load assumption in Eq. (7.3) can still be used by adopting the *Palmgren-Miner rule* (Miner, 1945; Kauzlarich, 1989). The idea is to calculate the accumulated damage D_{tot} after N_A cycles as follows:

$$D_{tot} = \sum_{i=1}^{N_A} \frac{n_i}{N(A_i)} \tag{7.4}$$

$$= \sum_{i=1}^{N_A} \frac{n_i}{\frac{K}{A_i^m}} \tag{7.5}$$

$$= \sum_{i=1}^{N_A} \frac{n_i A_i^m}{K} \ , \tag{7.6}$$

where n_i denotes the number of cycles that a periodic load with amplitude A_i is applied. Notice that Eq. (7.3) was used to express the expected number

of cycles $N(A_i)$ when the load amplitude is A_i in Eq. (7.5). In practice A_i and n_i are obtained using *rainflow counting* (Rychlik, 1987; ASTM E1049-85, 2017). The accumulated damage D_{tot} is used to determine whether the component will fail given a load time-series, which is the case for $D_{tot} > 1$.

In the context of wind turbines, the so-called *damage equivalent load* (DEL) is commonly used to quantify the accumulated fatigue. The DEL corresponds to the constant amplitude of a hypothetical periodic load that generates the same damage level as the one obtained from Eq. (7.6) within N_{eq} cycles. This is expressed as follows:

$$\frac{N_{eq}}{N(DEL)} = D_{tot} = \sum_{i=1}^{N_A} \frac{n_i A_i^m}{K} \tag{7.7}$$

$$\frac{N_{eq}}{\frac{K}{\mathrm{DEL}^m}} = \sum_{i=1}^{N_A} \frac{n_i A_i^m}{K} \tag{7.8}$$

$$\mathrm{DEL} = \sqrt[m]{\sum_{i=1}^{N_A} \frac{n_i A_i^m}{N_{eq}}}\,. \tag{7.9}$$

7.2.4 *Computational model summary*

By wrapping up the various steps of the simulation process, we are considering the data-driven scenario of having access to an experimental design containing wind speed realisations and the corresponding damage-equivalent or extreme loads on various components of the turbine. In total, $5,000$ samples are available which correspond to approximately 35 days of continuous inflow wind measurements.

In the simplest scenario, each of the input samples in $\mathcal{X} = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\right\}$ corresponds to the u component of a 10-minutes time series wind speed at hub height with sampling frequency of $20\,Hz$. The resulting random input vector $\mathbf{X} \in \mathbb{R}^M$ has dimensionality $M = 12,000$. When taking into account all the components of the wind vector, the dimensionality of $\mathbf{X}$ is tripled to $M = 36,000$. The outputs of interest $\mathcal{Y} = \mathcal{M}(\mathcal{X})$ are calculated based on post-processing the results of the aero-servo-elastic simulation of

TABLE 7.3: The outputs of the wind turbine computational model.

Symbol	Type	Load
Y_1	DEL	Blade 1 flap-wise bending moment at root
Y_2	DEL	Fore-aft bending moment at tower top
Y_3	Peak load	Fore-aft bending moment at tower top

the baseline 5MW wind turbine using OpenFAST. The model response $\mathbf{Y}$ consists of the three quantities of interest reported in Table 7.3.

7.3 DATA-DRIVEN FATIGUE AND PEAK LOAD PREDICTORS

7.3.1 *Methodology*

The goal in this section is to construct a surrogate model that provides predictions of the quantities of interest in Table 7.3, as a function of 10 minute measurements of the wind speed at hub height. Although the horizontal wind vector component u is expected to have the largest impact on the resulting loads, it is worth investigating whether the predictive performance of the resulting surrogate improves when all the wind vector components are taken into account. This poses additional challenges in terms of how to properly treat multiple structured inputs within a single input vector.

The surrogate modelling technique of choice is sparse polynomial chaos expansions (PCE), introduced in Section 4.3. The construction of such a surrogate that operates on an input space of dimensionality $\mathcal{O}\left(10^4\right)$ is prohibitive (see Section 5.1). Moreover, considering the quasi-periodicity of the loads, one should avoid to operate directly on the time domain. Instead, other representations, such as the frequency spectrum (Proakis and Manolakis, 1996) of the inflow wind time series, may provide a more compact and meaningful representation of the inputs with respect to the outputs of interest (functions of the loads). Using machine learning terminology, it is preferred to extract several *features* from the input time series that are as informative and discriminative as possible, rather than directly processing the raw input time-series.

To do so, we propose a solution that is based on a compression scheme of the form

$$g(\mathbf{X};\mathbf{w}) = \left(g_{KPCA} \circ g_{TSFRESH}\right)(\mathbf{X};\mathbf{w}), \tag{7.10}$$

where g_{KPCA} refers to kernel PCA (KPCA see Section 3.4) and $g_{TSFRESH}$ to a combination of several digital signal processing algorithms each extracting potentially meaningful, features out of the input time series using the PYTHON package "Time Series Feature Extraction on basis of Scalable Hypothesis tests", abbreviated as TSFRESH (Christ et al., 2018).

The transform $g_{TSFRESH}$ is of the form $g_{TSFRESH}(\mathbf{X}) = \{A_1(\mathbf{X}), \ldots, A_{M_A}(\mathbf{X})\}$, *i.e.* each feature $A_i(\mathbf{X})$ corresponds to a different operation on $\mathbf{X}$. In total, $M_A = 698$ features are computed, each corresponding to metrics such as summary statistics (mean, variance, minimum/maximum value, *etc.*), Fourier and Ricker wavelet coefficients (Ricker, 1944). The complete list of the extracted features can be found in Appendix C.3. Note that the final number of features $M_A = 698$ differs from the one listed in Table C.4 (788). A total of 90 features were removed due to having zero variance, *i.e.* a constant value over the entire dataset.

Note that feature extraction, such as $g_{TSFRESH}$, may not fulfil the purpose of dimensionality reduction adequately. In this application, indeed $M_A < M$ but it is still challenging to construct a surrogate with $M_A = 698$ inputs. Hence, we chose KPCA as a subsequent compression step (g_{KPCA} in Eq. (7.10)) to finally reduce the input space to a manageable size, $\mathcal{O}\left(10^{0-1}\right)$. However, the introduction of the $g_{TSFRESH}$ transform has numerous advantages compared to the direct application of out-of-the-self compression techniques, like KPCA, on $\mathbf{X}$: (i) It provides statistically more significant features of each input sample compared to the raw input components in the time domain (Christ et al., 2018), (ii) the methodology is still applicable in case of time series inputs of varying size, (iii) it improves the computational efficiency in further compression steps (increasingly important for larger M), and, (iv) each input time series (structured) is transformed to an unstructured vector making it straightforward to combine multiple such inputs and/or also include separate unstructured ones.

Recall from Section 3.4 that the g_{KPCA} transform in Eq. (7.10) is of the form $g_{KPCA}(\cdot;\mathbf{w})$, where $\mathbf{w}$ includes the kernel parameters (see Table 3.1) and the, unknown, reduced space dimension m. The calibration

of $\mathbf{w}$ is done using the DRSM algorithm that is described in Section 5.3, which optimises them in such a way that the surrogate model performance is optimal. DRSM is applied for each of the model responses $\mathbf{Y} = \{Y_1, Y_2, Y_3\}$ from Table 7.3, independently. Hence, a separate PCE surrogate $Y_i = \widehat{\mathcal{M}}^{(i)}\left(\mathbf{Z}; \boldsymbol{\theta}^{(i)}\right)$ $i = 1, \dots, 3$ is computed for each of them. The PCE of each output Y_i differs as dictated by the parameters $\boldsymbol{\theta}^{(i)}$ that correspond to the respective expansion coefficients (see Eq. (4.34)). The same applies for the compression scheme $\mathbf{Z} = g(\mathbf{X}; \mathbf{w}^{(i)})$ $i = 1, \dots, 3$, *i.e.* different KPCA parameters $\mathbf{w}^{(i)}$ are calculated for each output Y_i. Both parameter vectors $\mathbf{w}^{(i)}$ and $\boldsymbol{\theta}^{(i)}$ are jointly calibrated by minimising the objective function in Eqs. (5.6) and (5.7).

We also consider a modified version of the compression scheme in Eq. (7.10), that reads:

$$g(\mathbf{X}; \mathbf{w}) = \left(g_{KPCA} \circ g_{PCA} \circ g_{TSFRESH}\right)(\mathbf{X}; \mathbf{w}), \tag{7.11}$$

where g_{PCA} corresponds to an intermediate dimensionality reduction step using principal component analysis (PCA - see Section 3.2). The main motivation behind the addition of g_{PCA} lies on the observation that several features that $g_{TSFRESH}$ produces are highly correlated to each other (*e.g.* the features that correspond to different coefficients of the discrete Fourier transform in Table C.4). This is exploited by g_{PCA} that produces an intermediate reduced space of size $\mathcal{O}\left(10^{1-2}\right)$ with uncorrelated components (for proof of the decorrelation property of PCA see *e.g.* Bishop, 2006). The g_{KPCA} transform is still used at the end of the chain, with the added flexibility to use anisotropic kernels (such as the Gaussian in Table 3.1). This is highly challenging to do in the case of Eq. (7.10), because one kernel parameter should be calibrated for each of the 698 (or 3×698 if all wind components are processed) features that $g_{TSFRESH}$ produces.

A sketch of the various compression schemes that are compared during the DRSM phase of this study is shown in Figure 7.3, each denoted by a letter (A to D). The compression steps are sketched as boxes connected with arrows that indicate the order of each step and a number on top that refers to the dimensionality of the variable that enters or exits each transform. In setups A, B and C, a polynomial kernel is used in KPCA, whereas in setup D the anisotropic Gaussian kernel is used instead. The DRSM algorithm is used to tune only the KPCA parameters in each case, as well as the reduced space

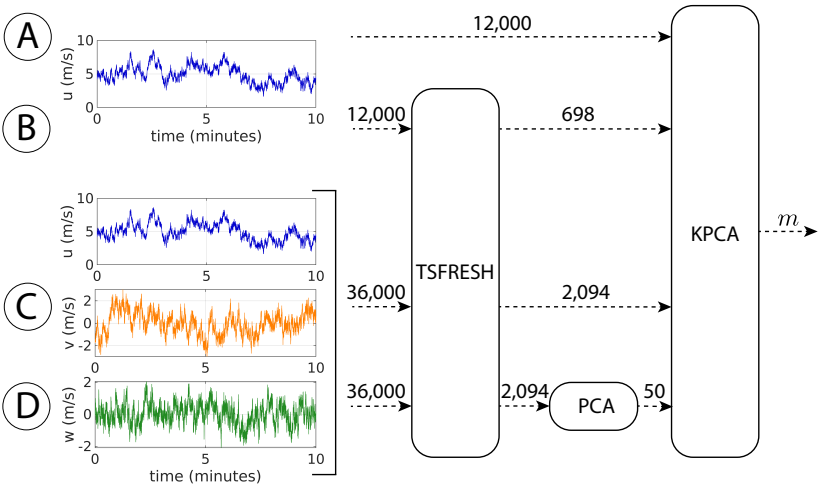


FIGURE 7.3: Illustration of the different setups that are used to compress the inflow wind time series.

dimension m but not any parameters related the the $g_{TSFRESH}$ transform in Eqs. (7.10) and (7.11) and the g_{PCA} transform in Eq. (7.11). Setup A refers to the simple case of directly applying KPCA compression in the time domain representation of the horizontal wind component u. Setups B and C correspond to the transformation in Eq. (7.10), considering either the u component only (setup B) or all three of them (setup C). Finally, setup D corresponds to the transformation in Eq. (7.11) and, similarly to setup C, takes into account the full inflow wind vector.

7.3.2 *Surrogate model performance evaluation*

To evaluate the proposed methodology, the 5,000 samples that are available, are randomly split into the mutually exclusive sets $\{\mathcal{X}, \mathcal{Y}\}$ with $N = 1,000$ samples used for calculating the surrogate (training set) and $\{\mathcal{X}_v, \mathcal{Y}_v\}$ with $N_v = 4,000$ samples, used for validation (validation set). In a practical setting, where the end-goal would be to deploy the trained surrogates one should use all the samples available for training them. This is further motivated by our findings in Section 5.4 where the leave-one-out error was consistently providing robust estimates of the generalisation error of surrogates, hence there is no need for training and validation splits. The

splitting of the data here is done purely for visualisation purposes (true versus predicted responses).

The random splitting of the samples into the training and the validation set is repeated 10 times. A simplifying assumption is made about the optimal reduced space dimension $\hat{m}$ being constant among the different splits. As such, it is only estimated once (from the first split) whereas the rest of the dimensionality reduction parameters are re-evaluated for each split. The leave-one-out (LOO) error of the PCE surrogate (see Eq. (4.45)), denoted by ε_{LOO}, is used to determine $\hat{m}$ because it is considered a robust estimator of the generalisation error.

The results of the $\hat{m}$ calibration process for each of the compression setups (A-D) and each of the model responses are shown in Table 7.4. Each group of four rows corresponds to one of the responses, Y_1 to Y_3. For each group, the DRSM algorithm is executed four times, each corresponding to the calculation of the optimal $\hat{m}$ and KPCA parameters using one of the four compression setups. The reported value of ε_{LOO} is the minimum value that was achieved with the respective compression setup using the $\hat{m}$ value listed next to it.

TABLE 7.4: Optimal reduced space dimension and corresponding leave-one-out error of the PCE surrogate for each model output using various compression setups.

Output	Comp. setup	$\hat{m}$	ε_{LOO}
Y_1	A	20	0.0325
	B	25	0.0332
	C	25	0.0233
	D	25	0.0230
Y_2	A	25	0.0576
	B	25	0.0189
	C	20	0.0207
	D	10	0.0239
Y_3	A	2	0.0759
	B	10	0.1269
	C	25	0.0675
	D	20	0.0945

For each of the 10 splits, the PCE surrogate is evaluated in terms of its predictive performance on the validation set, which serves as an estimator

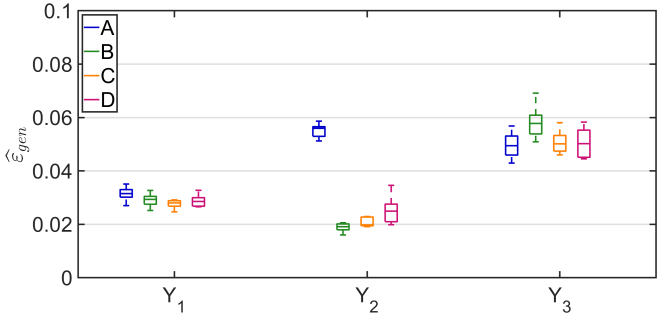


FIGURE 7.4: Generalisation error estimates for each model output using various compression schemes.

of its generalisation error, hence denoted by $\widehat{\varepsilon}_{gen}$. It reads (reminder of Eq. (4.42)):

$$\widehat{\varepsilon}_{gen} = \frac{\sum_{i=1}^{N_v} \left(\mathcal{M}(\mathbf{x}^{(i)}) - \widehat{\mathcal{M}}(\mathbf{x}^{(i)})\right)^2}{\sum_{i=1}^{N_v} \left(\mathcal{M}(\mathbf{x}^{(i)}) - \widehat{\mu}_y\right)^2},$$

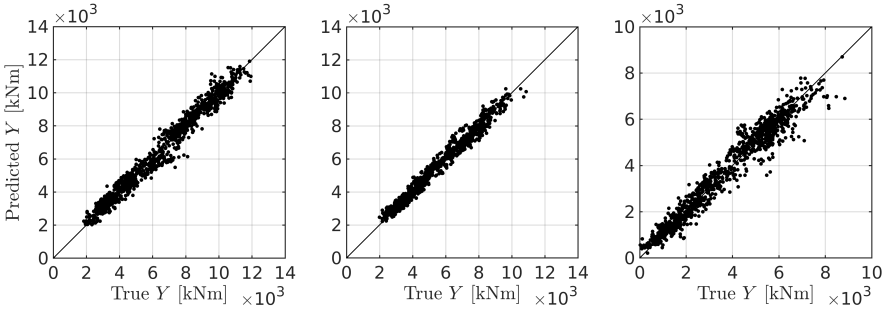
where $\widehat{\mu}_y = \frac{1}{N}\sum_{i=1}^{N_v} \mathcal{M}(\mathbf{x}^{(i)})$ is the sample mean of the validation set responses and $\widehat{\mathcal{M}}(\mathbf{x}^{(i)})$ is used in place of $\widehat{\mathcal{M}}(\mathbf{x}^{(i)}; \boldsymbol{\theta})$ to simplify the notation.

The value of $\widehat{\varepsilon}_{gen}$ of each surrogate over the 10 splits is visualised in Figure 7.4 using box plots. The central mark indicates the median, and the bottom and top edges of the box indicate the 25$^{\text{th}}$ and 75$^{\text{th}}$ percentiles, respectively. The whiskers extend to the most extreme data points up to 1.5 times the inter-quartile range above or below the box edges. Any sample beyond that range is considered an outlier and plotted as a single point. The value of the mean $\widehat{\varepsilon}_{gen}$ that each method achieves over the 10 splits is also given numerically in Table 7.5. Each box plot refers to the PCE that was calculated on the reduced space obtained using each of the compression setups that were previously described (see also Figure 7.3). The box plots that correspond to each of the three model responses are grouped and in each case their colour differs depending on the compression setup.

At a first glance, it is clear that different setups perform best depending on which model response is considered. Regarding the DEL of the flap-wise bending moment at the blade root (Y_1), the horizontal component of the

TABLE 7.5: Mean $\widehat{\varepsilon}_{gen}$ achieved by a PCE surrogate over 10 random splits of the training and validation set using various compression setups. The lowest value for each output is highlighted in bold.

Output	A	B	C	D
Y_1	0.0315	0.0292	**0.0276**	0.0292
Y_2	0.0551	**0.0190**	0.0212	0.0252
Y_3	**0.0497**	0.0584	0.0506	0.0505



(a) DEL of flap-wise bending moment at blade root (Y_1)

(b) DEL of fore-aft bending moment at tower top (Y_2)

(c) Peak fore-aft bending moment at tower top (Y_3)

FIGURE 7.5: Comparison between true and end predicted model responses (on the validation set).

inflow wind vector is the main driver of the output value, but taking into account the full vector (setups C and D) leads to a marginal improvement of the generalization error in the order of $3-7\%$. However, the additional information of the v and w wind components seems to have no contribution when the outputs Y_2 and Y_3 are considered instead, that correspond to the DEL and peak load of the fore-aft bending moment at the tower top. In both scenarios that a DEL is predicted (outputs Y_1 and Y_2), using the multi-step compression setups (B, C and D) consistently leads to superior results when compared to the setup A, that is directly applying KPCA on the wind time series. This is not the case for Y_3, where all setups show comparable performance.

Finally, Figure 7.5 provides a visualisation of the predictive performance of the best performing setup in Table 7.5 for each model response. Figure 7.5a

(resp. Figure 7.5b and Figure 7.5c) contains a scatter plot of the true values against the ones predicted by the PCE surrogate of the response Y_1 (resp. Y_2 and Y_3). For this comparison, only the samples of the validation set of one of the 10 splits of the previous analysis are considered. In accordance to the $\widehat{\varepsilon}_{gen}$ values in Table 7.5, the best predictor performance is achieved for Y_2 followed by Y_1 and Y_3. The DEL predictors (Y_1 and Y_2) appear unbiased with consistent performance across different regions. The peak load predictor (Y_3) also appears unbiased for the most part with a slight tendency to overestimate the damage in the lower tail region.

7.4 RESPONSE PDF ANALYSIS

The loads applied to various components of a wind turbine are uncertain due to the variability of the wind climate. Dealing with a high-dimensional input space that consists of one or more time series, introduces numerous challenges in quantifying those uncertainties. In this section, we showcase how such analyses can be enabled using the reduced dimension resampling (RDR) method proposed in Chapter 6, by approximating the full response PDF using Monte Carlo simulation.

The problem setting is the same as in the previous section, *i.e.* there is a limited number of observations $\{\mathcal{X}, \mathcal{Y}\}$ available, $5,000$ in total, and the goal is to estimate the response PDF as accurately as possible. The first step of the RDR approach (see Section 6.2.1), is to calculate the optimal compression scheme as well as the surrogate (PCE in this case) that operates in the reduced space, using the DRSM algorithm. This step has already been covered in the previous section. Based on the performance evaluation of the various compression setups in Table 7.5, we pick the best performing one for each of the three responses and repeat the training process using the entire dataset this time (instead of using only $1,000$ samples). Splitting the samples in training and validation sets would not be as informative in this case. Using such a small number of samples ($\mathcal{O}\left(10^3\right)$) as a validation set, *i.e.* to determine the reference values of the response statistics, could lead to misleading results. ´

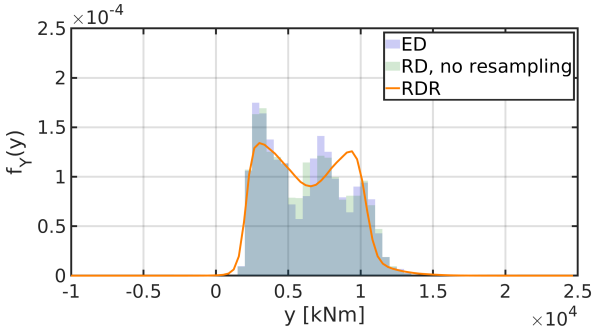
The next step is to approximate the joint PDF of the reduced space $f_{\mathbf{Z}}(\mathbf{z})$. This process is repeated three times, once for each model output, because a different reduced space is determined for each of them. As discussed in Section 6.2.2, this approximation decomposes to: (i) estimating the marginal

distribution of each component Z_i, $i = 1, \ldots, m$ using kernel smoothing, and, (ii) fitting a canonical vine copula (C-vine) to model the dependency structure. Having such a probabilistic input model enables the generation of an arbitrary number of samples in the reduced space that mimics the statistics of the experimental design (see Section 6.2.3).

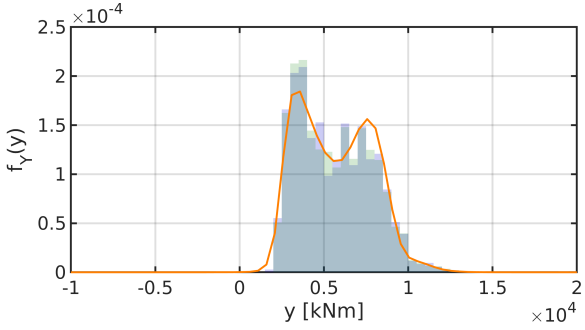
Next, we estimate the full response PDF of each output of interest from Table 7.3 using Monte Carlo simulation (see Section 6.2.4). This is achieved with RDR by generating 10^6 artificial samples $\mathcal{Z}' \sim \widehat{f}_{\mathbf{Z}}(\mathbf{z})$ and then computing the corresponding model responses using the PCE surrogate. Out of the four compression setups we used the one that results in the best performing PCE for each output, highlighted in Table 7.5.

The response PDF estimates $\left\{f_{Y_i}(y_i)\,,\ i = 1, 2, 3\right\}$ are visualised in Figure 7.6. Figure 7.6a (resp. Figures 7.6b and 7.6c) contain the results for the output Y_1 (resp. Y_2 and Y_3). Each panel contains: (i) a histogram generated from the available $\mathcal{Y}$ samples, (ii) a histogram generated from $\widehat{\mathcal{Y}} = \widehat{\mathcal{M}}(g(\mathcal{X}))$, *i.e.* by compressing the experimental design and estimating the approximate responses using the PCE surrogates, and, (iii) a curve that corresponds to the kernel-smoothing-based estimate of $f_{Y_i}(y_i)$ using the 10^6 samples that were generated by RDR.

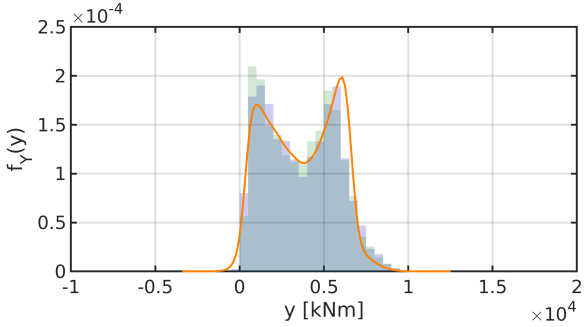
The estimates of f_{Y_1} (DEL of the flap-wise bending moment at the root of blade 1), show significant differences depending on the samples that are used. RDR identifies a bimodal distribution, whereas a trimodal distribution is observed in the experimental design. Recall that the approximation error of the response PDF by RDR is due to the input compression, surrogate modelling and resampling steps. To obtain a qualitative estimate of the error that is introduced by the input compression and the surrogate, we compare the two histograms in Figure 7.6a, because any difference between them is only due to those factors. The difference between the two histograms does not seem large enough to justify the different shape of the RDR-based estimate and the tri-modality is retained. This is an indicator that the resampling error may be the main cause of this discrepancy. Regarding the outputs Y_2 and Y_3, *i.e.* the DEL and peak of the bending moment at the top of the tower, RDR shows a refined representation of the two modes of each distribution. However, considering the Y_2 PDF, the second mode that RDR identifies is not clearly visible in the available samples.



(a) DEL of flap-wise bending moment at blade root (Y_1)



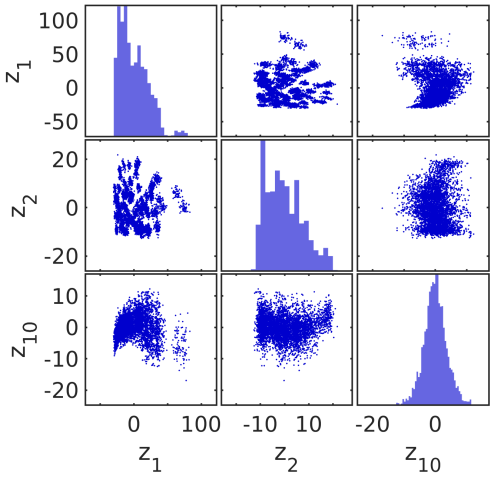
(b) DEL of fore-aft bending moment at tower top (Y_2)



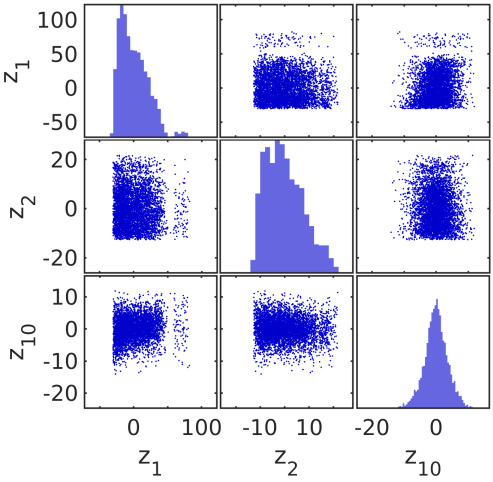
(c) Peak fore-aft bending moment at tower top (Y_3)

FIGURE 7.6: Estimates of the PDF's of the computational model responses.

The significant discrepancy between the experimental-design- and RDR-based estimates of f_{Y_1}, motivates a further investigation to identify the cause.



(a) Experimental design in reduced space



(b) Artificial samples by RDR

FIGURE 7.7: Scatter plots and marginal histograms of different samples in the optimal reduced space for Y_1. Only 3 out of the 20 components are shown.

To test the accuracy of the reduced space resampling step, we compare the scatter plots and marginal histograms of $\mathcal{Z} = g(\mathcal{X})$, *i.e.* the representation of

the experimental design in the reduced space, and of the artificial samples $\mathcal{Z}'$. Out of the 20 components of the reduced space, in Figure 7.7 we visualise a representative subset in terms of the dependency patterns that we observe, with a focus on the more challenging ones. Each of the figures contains 9 panels in a 3×3 grid. The panel located in the i-th row and j-th column, for $i \neq j$ shows a scatter plot of the i-th versus the j-th component of $\mathbf{Z}$ using either the $\mathcal{Z}$ samples (in Figure 7.7a) or the same number of samples from $\mathcal{Z}'$ (in Figure 7.7b). The panels on the diagonal ($i = j$) show the histogram of the respective component of $\mathbf{Z}$ instead. At a first glance, the marginal distributions appear to be captured well, especially for components Z_1 and Z_{10}. However, the dependency patterns show significant discrepancy between the true and the artificial samples. Considering the scatter plots of Z_1 versus Z_2, the training samples appear to form clusters around numerous regions while other portions of the domain are empty. This is a behaviour that cannot be captured with the copulas we used and the artificial samples show a more even coverage of the domain instead. Further investigation would be required to determine whether evenly covering the domain is realistic, in terms of the wind climate that corresponds to samples in various regions that are not covered by $\mathcal{Z}$. In addition, a dependency pattern like the one we observe between Z_1 (or Z_2) and Z_{10} indicate some form of functional relationship that also cannot be adequately captured by the copulas we used.

7.5 CONCLUSION

A wind turbine is a highly challenging system to design and operate. On the one hand, its static and dynamic behaviours are driven by widely varying load conditions, such as the wind climate, the uncertainty of which is non-trivial to quantify. On the other hand, quantities of interest such as the moments or forces on critical components, show a complex non-linear relationship with those loads, and therefore require aero-servo-elastic simulators that are costly to run.

In this work we focus on the estimation of fatigue, by means of damage equivalent loads, as well as peak loads on various components, given a time series of the inflow wind vector. We approach the problem from a purely data-driven perspective, where the entire analysis is based on a limited set of observations. The main challenge lies in the construction of a surrogate model given the high dimensionality of the input space ($\mathcal{O}\left(10^4\right)$).

We tackle this problem by compressing the input space using kernel PCA in such a way that the attached PCE surrogate has optimal generalisation performance.

Moreover, we expand this approach to more elaborate compression strategies that involve the extraction of several hundreds of features from each input time series. We also combine multiple input time series (in this case the three wind speed vector components) in a single unstructured input. The latter can either be directly processed by kernel PCA using a polynomial kernel (or other isotropic ones) or be further compressed and de-correlated using PCA, followed by kernel PCA with an anisotropic kernel.

Having access to such surrogates can be useful in various scenarios. For example, during the design process, they can substitute the computationally expensive aero-servo-elastic simulators after training them on a limited number of samples. After deploying the wind turbine in the field, the surrogate can also be used to process continuous inflow wind measurements and estimate the fatigue accumulation on different components.

We further exploited the trained surrogates by proceeding to the quantification of the uncertainty of the model responses (damage equivalent and peak loads). This was achieved by performing Monte Carlo simulation using the reduced dimension resampling technique, proposed in Chapter 6, that results in estimates of the full response PDF. We showed that by doing so, it is possible to obtain a refined representation of the PDFs compared to simply processing the limited amount of response measurements at our disposal. We also identified a limitation of the current approach, that is given by complex dependency structures in the reduced space. Therefore, the accuracy of this methodology strongly depends on the complexity of reduced representation of the input space. This finding motivates extensions of RDR that are further discussed in Section 8.2.

The proposed methodology can be directly applied to more complex scenarios that involve more structured (time series) inputs, as well as unstructured ones, by using the compression schemes in Eq. (7.10) or Eq. (7.11). It is noteworthy that there are only a few parameters that need to be tuned manually, related to the low- and high- fidelity surrogates the kernel of kernel PCA and, optionally, the optimisation algorithm of DRSM. Considering that the leave-one-out error of the surrogate provides a robust estimator of its generalisation performance, one could essentially automate the entire parameter

tuning process. To do so, simple grid search algorithms or more efficient Bayesian optimisation ones (see *e.g.* Snoek et al., 2012) could be used for finding the parameter set that minimises the leave-one-out error.

An extension to this work would be to capitalise on the estimated response PDFs and calculate the probability of the fatigue accumulation on different components exceeding the design limits in future time instances. Recall that each sample corresponds to the DEL over a fixed time interval (10 minutes in this work). Hence, failure in a specific future time instance corresponds to the probability that a fixed number of samples drawn from the response PDF, and properly summed to calculate the total DEL (see *e.g.* Berglind et al. (2016)), result in a total DEL larger that the limit value. However, it is important that the provided samples are representative of the wind climate over all the seasons.

8

CONCLUSIONS

8.1 SUMMARY

The present work aims to bridge the gap between the uncertainty quantification and machine learning communities in order to enable uncertainty quantification in engineering problems that present two characteristics: (i) a high dimensional input space, and, (ii) a purely data-driven context, which means that the system under investigation is only known through a limited number of available observations, without the possibility of enriching the existing dataset.

On the one hand, the machine learning community has proposed several techniques for compressing high dimensional spaces for a variety of applications (*e.g.* data compression, visualisation, image de-noising) but they typically do not take into account uncertainties. On the other hand, the uncertainty quantification community has contributed several methodologies for properly handling the underlying uncertainties, but they tend to scale unfavourably or even become intractable as the input dimensionality increases. In this thesis we tried to combine the "best of both worlds", by using machine learning techniques for input compression and surrogate modelling and uncertainty propagation techniques that have been developed primarily by the uncertainty quantification community.

As a first step, we identified the state-of-the-art of the two fundamental ingredients: dimensionality reduction and surrogate modelling in Chapters 3 and 4, respectively. Chapter 3 provides a literature review on the latest developments on dimensionality reduction techniques. The focus was given primarily to non-linear techniques and their association to classical and mostly linear techniques, namely principal component analysis and multidimensional scaling. Furthermore, we pointed out the strengths and weaknesses of each method and concluded that selecting a suitable dimensionality reduction method depends on several factors, such as the

complexity of the problem and the type of application (*e.g.* visualisation, data compression, image denoising).

In Chapter 4 we provided a review on the state-of-the-art in surrogate modelling and introduced two particular techniques that are used throughout this thesis, Gaussian process modelling (Kriging) and sparse polynomial chaos expansions. Although both methods are well-established within the field of uncertainty quantification, they suffer from the curse of dimensionality, *i.e.* cannot be directly applied to the high-dimensional problems that are of interest in this thesis.

In Chapter 5 we present a novel methodology for enabling surrogate modelling in problems with large input dimensionality. The proposed DRSM algorithm couples the input compression and surrogate modelling steps in such a way that the resulting performance of the surrogate model is optimal. The novelty of this algorithm lies in how the two stages are coupled into a single problem, for which dedicated solution strategies are proposed. This type of coupling allows the combination of various dimensionality reduction and surrogate modelling methods without having to tweak the dedicated optimisation algorithms on which each of them capitalises.

The performance of DRSM was tested on benchmark problems and compared against the classical approach of tuning the dimensionality reduction and surrogate modelling parameters sequentially. For this purpose we used kernel principal component analysis for dimensionality reduction and tested both Kriging and polynomial chaos expansions for surrogate modelling. The proposed DRSM methodology consistently showed superior performance in all the benchmarks. Furthermore, we demonstrated that the leave-one-out cross-validation error of the surrogate is a reliable estimator of the generalisation error. As such, it can be used as an objective for tuning the dimensionality reduction parameters, but also to assess the overall accuracy of the resulting surrogate.

In Chapter 6 we propose a workflow for data-driven uncertainty propagation in problems with high dimensional input spaces. In this emerging research field, one has to deal with several challenges. On the one hand, "data-driven" refers to the limited knowledge about the inner workings of the system under consideration, as well as the uncertainty of the input parameters (no prescribed probabilistic models are available). A limited set of input samples and the corresponding model responses is the only

available information about the system. On the other hand, the high input dimensionality renders various state-of-the-art uncertainty quantification techniques for surrogate modelling and inference non-applicable.

The workflow that we propose, called reduced dimension resampling (RDR), uses as a starting point the reduced space that we obtain by using the DRSM algorithm. Having access to a compressed representation of the input space of manageable size, we used well-established techniques for resampling (briefly presented in Chapter 2). In addition, we adopted the copula formalism to model the dependency structure of the inputs, and in particular canonical vine copulas, because they allow for data-driven, computationally efficient and scalable inference of their parameters. Eventually, the response PDF is estimated by Monte Carlo simulation, *i.e.* by propagating the newly generated samples of the reduced input space through the surrogate model, provided by DRSM. By using the proposed methodology on benchmark applications, we demonstrated that RDR indeed improves our knowledge about the output uncertainty in terms of the overall shape of the PDF and its empirical quantiles.

Finally, in Chapter 7 we show how the methods presented in this thesis can be applied in a realistic engineering application related to the structural health monitoring of wind turbines. The goal is to estimate the fatigue accumulation and peak loads, as well as their uncertainty, on various components of a wind turbine, given the inflow wind speed over 10 minute time intervals. We do so by processing a limited amount of observations that are generated by specialised software. By applying the techniques introduced in Chapter 5, we calculated polynomial chaos expansions surrogates that operate on reduced input spaces of dimension $2 - 25$ whereas the physical space is $36,000$-dimensional.

For this particular class of problems, where the input contains one or more time series, we proposed several custom compression setups that involve a feature extraction step before applying a standard compression algorithm such as kernel PCA. The main motivation for working on features of the input time series (from simple metrics to Fourier and wavelet coefficients) is that they provide a more compact representation of the input space. There are also practical advantages. Structured inputs are transformed to unstructured, allowing us to easily combine features from multiple time series in a single input vector. In addition, the same workflow is applicable

even in cases where the length of the time series is not identical (*e.g.* due to variation in the sampling frequency).

Furthermore, the uncertainty of the quantities of interest was quantified using the RDR methodology introduced in Chapter 6. This led to a refined representation of the response PDFs in most cases. In one case the proposed methodology under-performed and the root cause was the dependency structure of some components in the reduced space that could not be adequately captured with copulas. Nevertheless, we showed how one can anticipate for such errors even in the absence of a validation set by inspecting the resampling and surrogate model performance.

Overall, this manuscript provides a new set of tools that enable the application of uncertainty quantification techniques in a wide class of problems, for which it was initially not possible. This has strong practical implications considering the numerous relevant problems nowadays in *e.g.* structural health monitoring, earthquake engineering, weather forecasting, hydrogeology and control engineering, where the input space is high-dimensional (*e.g.* time series or image inputs).

As a closing remark, the UQLab software framework played a crucial role in the deployment of the proposed algorithms. Having a single framework (as opposed to different methodologies provided by different packages even in different programming languages) that allowed to plug-in various state-of-the-art surrogate modelling and probabilistic model inference techniques resulted in a significantly reduced development time. Moreover, having access to highly computationally optimised routines for training each surrogate made the entire idea of DRSM feasible. It allowed to generate hundreds or thousands of low-fidelity surrogates during each DRSM optimisation within a feasible time frame of minutes to a few hours, depending on the problem.

8.2 OUTLOOK

8.2.1 *DRSM algorithm*

Throughout this thesis we focused on high-dimensional input spaces. However, it is of practical interest to expand the proposed methodology to high dimensional output spaces as well, *e.g.* for problems with time series

responses. An additional compression step would be required then for the output space. In principle, the DRSM algorithm could be directly applied with slight modifications to also include the parameters of the output compression scheme in the outer optimisation loop in Eq. (5.6). Nevertheless, special care needs to be taken in the compression technique that we use for the output space. In contrast to the case of compressing the input space where the inverse transformation (from the reduced to the physical space) was not required this is not true for the output compression, because we would have to eventually map back the responses that the surrogate predicts to the actual output space.

In this work we applied the concept of DRSM with techniques that become computationally intractable in the presence of a large number of samples in the experimental design ($\mathcal{O}\left(10^6\right)$ or more). This is rather a limitation of the input compression and surrogate modelling techniques and not of DRSM. Regarding the input compression scheme, autoencoders (presented in Section 3.5) could be used instead of kernel PCA due to their low memory requirements. Memory limitations also apply to the surrogate modelling techniques presented in Chapter 4. One way around this issue is to exploit potential redundancies in the provided samples by first calculating a surrogate using a small subset of them and then iteratively enrich the experimental design only in regions where the observed error is large. The key difference of the DRSM implementation in such problems is the optimisation algorithm that should be used. Global optimisation or more generally methods that require the processing of the entire dataset for each objective function evaluation become computationally inefficient. Instead, local (gradient-based) techniques should be preferred, such as stochastic gradient descent, because they only process small batches of samples in each iteration.

The concept of DRSM could be further exploited to deal with problems to which classical surrogate modelling techniques can be applied with no computational issues, but under-perform due to the complexity of the input-output map. From a machine learning standpoint, techniques like kernel PCA are still applicable in this setting for the purpose of feature engineering, *i.e.* producing transformations of the input space that hopefully make the input-output map less complex to model. No modifications of the proposed algorithm are necessary to deal with this class of problems apart from alleviating the constraint that the transformed input space has reduced

dimension. On the contrary, one could allow for larger dimensions than the physical space and let DRSM determine the optimal intrinsic dimensionality value.

8.2.2 *Uncertainty quantification in high dimensional input spaces*

The methodology that was proposed in this thesis for uncertainty propagation in high-dimensional input spaces laid the ground work for numerous future extensions. Having access to a surrogate model and a probabilistic representation of the (reduced) input space, this challenging class of problems can be approached using standard uncertainty quantification techniques. For example, an extension of practical interest in engineering applications is the calculation of the probability of failure of a system, also known as reliability analysis (Ditlevsen and Madsen, 1996; Melchers, 1999).

The limitation of the reduced dimension resampling technique that we encountered in Section 7.4 reminisces our preliminary findings in Section 5.2. Similar to the way that input compression may introduce a complex input-output map, it may also result in a space with a complex joint PDF. A future research direction would be to penalise a complex joint PDF during the DRSM optimisation, using a suitable regularisation term in the outer loop optimisation in Eq. (5.6). Alternatively, a new reduced space could be determined only for resampling purposes. In that case, dimensionality reduction methods that provide an accurate inverse transform from the reduced to the physical space should be preferred. According to this workflow, new samples are drawn in this new reduced space, mapped back to the physical space and then compressed again in the former reduced space that the surrogate operates in order to finally calculate the model response estimates. In both extensions, an open question is what quantitative criterion should be used to describe the "goodness" of a space with respect to the complexity of its joint PDF.

A

THEORETICAL REMARKS

A.1 THE RELATIONSHIP BETWEEN PCA AND KPCA WITH LINEAR KERNEL

Consider the PCA-based dimensionality reduction $\mathbf{x} \in \mathbb{R}^M \mapsto \mathbf{z} \in \mathbb{R}^m$. As discussed in Section 3.2, $\mathbf{z}$ is calculated as follows:

$$\mathbf{z} = \mathbf{x}\,\boldsymbol{V}, \tag{A.1}$$

where $\boldsymbol{V} \in \mathbb{R}^{M \times m}$ is the collection of the m eigenvectors of $\boldsymbol{C} = \operatorname{cov}[\mathcal{X}]$ and $\mathcal{X} \in \mathbb{R}^{N \times M}$ is the experimental design. Compared to the more general formulation in Eq. (3.8), in Eq. (A.1) we assume zero mean $\boldsymbol{\mu}_{\mathbf{X}}$ for simplicity.

Next, consider the kernel PCA mapping $\mathbf{x} \in \mathbb{R}^M \mapsto \boldsymbol{q} \in \mathbb{R}^m$ using the linear kernel function:

$$\kappa\left(\boldsymbol{x}, \boldsymbol{x}'\right) = a\,\boldsymbol{x}^\top \boldsymbol{x}' + b. \tag{A.2}$$

It is straightforward to show that the following transformation is equivalent to the linear kernel in Eq. (A.2):

$$\Phi(\mathbf{x}) = \left\{\sqrt{b}, \sqrt{a}\,x_1, \ldots, \sqrt{a}\,x_M\right\}^\top, \tag{A.3}$$

because $\kappa\left(\boldsymbol{x}, \boldsymbol{x}'\right) = \Phi(\mathbf{x})^\top \Phi(\mathbf{x}')$. A sample $\boldsymbol{q}$ in the reduced space is calculated as follows (see Section 3.4):

$$\boldsymbol{q} = \Phi(\mathbf{x})^\mathsf{T} \boldsymbol{V}_{\mathcal{H}}, \tag{A.4}$$

where $\boldsymbol{V}_{\mathcal{H}}$ is the collection collection of m eigenvectors of $\boldsymbol{C}_{\mathcal{H}} = \text{cov}\left[\Phi(\mathcal{X})\right]$. Notice that in case of $a = 1$ and $b = 0$, from Eqs. (A.1), (A.4) follows that $\mathbf{z} = \boldsymbol{q}$.

The covariance matrix $\boldsymbol{C}_{\mathcal{H}}$ can be expressed as:

$$\boldsymbol{C}_{\mathcal{H}} = \begin{bmatrix} 0 & \dots & 0 \\ \vdots & & \\ 0 & & \sqrt{a}\,\boldsymbol{C} \end{bmatrix}. \tag{A.5}$$

Hence, excluding the eigenvector that corresponds to the zero eigenvalue, it is straightforwards to show that

$$\boldsymbol{V}_{\mathcal{H}} = \begin{bmatrix} 0 & \dots & 0 \\ & \boldsymbol{V} & \end{bmatrix}. \tag{A.6}$$

Based on Eqs. (A.3) and (A.6), Eq. (A.4) can be written as follows:

$$\boldsymbol{q} = \begin{bmatrix} \sqrt{b} & \sqrt{a}\,\mathbf{x}^{\top} \end{bmatrix} \begin{bmatrix} 0 & \dots & 0 \\ & \boldsymbol{V} & \end{bmatrix} \tag{A.7}$$

$$= \sqrt{a}\mathbf{x}^{\top}\boldsymbol{V} \tag{A.8}$$

$$= \sqrt{a}\mathbf{z} \quad \text{(from Eq. (A.1))} \tag{A.9}$$

Therefore, the dimensionality reduction using kernel PCA with a linear kernel provides a scaled version of standard PCA and the constant b has no effect.

A.2 ESTIMATING THE SIMILARITY OF PROBABILITY DISTRIBUTIONS USING THE JENSEN-SHANNON DIVERGENCE

The Jensen-Shannon divergence of two continuous PDFs, $p(x)$ and $q(x)$, reads (Lin, 1991):

$$JS(p \parallel q) = \frac{1}{2} KL(p \parallel \frac{1}{2}(p+q)) + \frac{1}{2} KL(q \parallel \frac{1}{2}(p+q)), \tag{A.10}$$

where $KL(p \parallel q)$ denotes the Kullback-Leibler (KL) divergence that is defined as follows:

$$KL(p \parallel q) = \int_{-\infty}^{+\infty} p_X(x) \log\left(\frac{p_X(x)}{q_X(x)}\right) dx. \tag{A.11}$$

A sample-based approximation of the RHS of Eq. (A.11) is obtained by:

$$KL(p \parallel q) \approx \sum_{i=1}^{n_b} \tilde{p}_i \log\left(\frac{\tilde{p}_i}{\tilde{q}_i}\right), \tag{A.12}$$

where $\tilde{p}_i$ (resp. $\tilde{q}_i$) denotes normalised histogram value of the samples from $p_X(x)$ (resp. $q_X(x)$) on the i-th bin and n_b denotes the number of bins. The JS divergence is a similarity measure for probability distributions that is based on the KL divergence with some notable differences, including that it is symmetric (trivial to show that $JS(p \parallel q) = JS(q \parallel p)$ from Eq. (A.10)) and it has always a finite value (see Lin, 1991 for proof).

A.3 EMPIRICAL STATISTICS ESTIMATORS

The goal of this section is to outline standard techniques for calculating the estimators of the mean, variance and quantiles of a random variable X from the samples $\mathcal{X}$. Those estimators are used by the reduced dimension resampling technique as discussed in Section 6.2.4.

The moments of X can be approximated by their empirical estimators:

$$\widehat{\mu}_X = \frac{1}{N} \sum_{i=1}^{N} x^{(i)}, \tag{A.13}$$

for the mean response, and:

$$\widehat{\sigma}_X^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(x^{(i)} - \widehat{\mu}_X\right)^2, \tag{A.14}$$

for the variance of the response. Similar estimates can be calculated for higher order moments. The Monte-Carlo-based moment estimates are essentially random variables, that are asymptotically Gaussian-distributed based on the central limit theorem. The variance of the mean and variance estimator can be analytically determined as:

$$\mathrm{Var}\left[\widehat{\mu}_X\right] = \frac{\sigma_X^2}{N} \tag{A.15}$$

$$\mathrm{Var}\left[\widehat{\sigma}_X^2\right] = \frac{\sigma_X^4}{N}\left(\kappa - 3 + \frac{2N}{N-1}\right) \tag{A.16}$$

where κ corresponds to the kurtosis of X, that is the 4-th standardised moment, *i.e.* $\kappa \overset{\text{def}}{=} \mathbb{E}\left[\left(\frac{X-\mu_X}{\sigma_X}\right)^4\right]$. Those quantities can be useful in practice because they allow the calculation of confidence intervals on the estimated values. Provided that N is sufficiently large, the $1-\alpha$ confidence interval on $\widehat{\mu}_X$ reads:

$$\widehat{\mu}_X - u_{\alpha/2}\,\mathrm{Var}\left[\widehat{\mu}_X\right]/\sqrt{N-1} \leq \mu \leq \widehat{\mu}_X + u_{\alpha/2}\,\mathrm{Var}\left[\widehat{\mu}_X\right]/\sqrt{N-1}, \tag{A.17}$$

where $u_{\alpha/2} = -\Phi^{-1}\left(1-\alpha/2\right)$. A similar expression to the one in Eq. (A.17) can be used to calculate the confidence intervals on the variance estimator.

One way of obtaining robust statistics of X, beyond moments, is to use the *empirical quantiles*. The quantile $Q(p)$ of a distribution is defined as:

$$Q(p) = F_X^{-1}(p) = \inf\left\{x : F_X(x) \geq p\right\}, \tag{A.18}$$

where $F_X(x)$ denotes the CDF of X. In the absence of $F_X(x)$, one can estimate empirical quantiles from the samples $\mathcal{X}$. Let $\tilde{\mathcal{X}} = \left\{x^{[1]}, \ldots, x^{[N]}\right\}$

denote the ordered statistics of $\mathcal{X}$, *i.e.* $x^{[1]} \leq x^{[2]} \leq \ldots \leq x^{[N]}$. A *plotting position* p_i is attached to each sample $x^{[i]}$, $i = 1, \ldots, N$ with value:

$$p_i = \frac{i}{N+1}, \tag{A.19}$$

with the property $\widehat{Q}(p_i) = x^{[i]}$, *i.e.* the empirical quantile of order p_i is the sample value $x^{[i]}$. To compute empirical quantiles of other orders $\alpha \in [1/(N+1), N/(N+1)]$ with $\alpha \neq p_i$, $i = 1, \ldots, N$, linear interpolation is used:

$$\widehat{Q}(\alpha) = x^{[i]} + \frac{x^{[i+1]} - x^{[i]}}{p_{i+1} - p_i}\alpha, \tag{A.20}$$

after finding the index i such that $p_i < \alpha < p_{i+1}$.

B

THE GAUSSIAN PROCESS MODELLING MODULE IN UQLAB

B.1 INTRODUCTION

Uncertainty quantification (UQ) through computer simulation is an interdisciplinary field that has seen a rapid growth in the last decades. Broadly speaking, it aims at (i) identifying and quantifying the uncertainty in the input parameters of numerical models of physical systems, and (ii) quantitatively assessing its effect on the model responses. Such a general formulation comprises a number of applications, including structural reliability (Lemaire, 2009), sensitivity analysis (Saltelli et al., 2000), reliability-based design optimisation (Tsompanakis et al., 2008) and Bayesian techniques for calibration and validation of computer models (Dashti and Stuart, 2017).

Due to the high cost of repeatedly evaluating complex computational models, analyses with classical sampling techniques such as Monte Carlo simulation are often intractable. In this context, meta-modelling techniques (also known as surrogate modelling) allow one to develop fast-to-evaluate surrogate models from a limited collection of runs of the original computational model, referred to as the experimental design (Santner et al., 2003; Fang et al., 2005; Forrester et al., 2008). Popular surrogate modelling techniques include Kriging (Sacks et al., 1989), polynomial chaos expansions (Ghanem and Spanos, 1991; Xiu and Karniadakis, 2002) and support vector regression (Vapnik, 1995).

Kriging is a surrogate modelling technique first conceived by Krige (1951) in the field of geostatistics and later introduced for the design and analysis of computer experiments by Sacks et al. (1989) and Welch et al. (1992). The potential applications of Kriging in the context of uncertainty quantification range from basic uncertainty propagation to reliability and sensitivity analysis (Marrel et al., 2008; Echard et al., 2011; Iooss and Lemaître, 2015b;

Le Gratiet et al., 2016). By exploiting the Gaussian properties of a Kriging surrogate output, additional applications include adaptive, surrogate model-based optimisation (see *e.g.*Moustapha et al. (2016)) and Bayesian calibration of computer models (see *e.g.*Bachoc et al. (2014)). Although in its standard form Kriging is a stochastic interpolation method, certain extensions have been proposed for dealing with noisy observations. Such extensions have been of particular interest to the machine learning community and they are commonly referred to as *Gaussian process regression* (Rasmussen and Williams, 2006).

A number of dedicated toolboxes are readily available for calculating Kriging surrogate models. Of interest to this review is general purpose software not targeted to specific Kriging applications, because they are typically limited to two or three dimensional problems (see *e.g.*gslib (Deutsch et al., 1992)). Within the R community one of the most comprehensive and well-established Kriging packages is arguably DiceKriging, developed by the DICE consortium (Roustant et al., 2012). This set of packages provides Kriging meta-modelling as part of a framework for adaptive experimental designs and Kriging-based optimisation based on the packages DiceDesign and DiceOptim (Dupuy et al., 2015; Picheny et al., 2016). SCIKIT-LEARN provides a PYTHON-based, machine-learning-oriented implementation of Gaussian processes for regression and classification (Pedregosa et al., 2011). Alternatively, PyKriging (Paulson and Ragkousis, 2015) offers a Kriging toolbox in PYTHON that offers basic functionality with focus on user-friendliness. Gpy (GPy, 2012) offers a Gaussian process framework with focus on regression and classification problems. Within the MATLAB programming language the first Kriging toolbox with widespread use was DACE (Lophaven et al., 2002). DACE was later extended to ooDACE (Couckuyt et al., 2014), an object-oriented Kriging implementation with a richer feature set. Small Toolbox for Kriging (Bect et al., 2014) offers an alternative Kriging implementation that is mainly focused on providing a set of functions for Kriging surrogate modelling and design of experiments. GPML (Rasmussen and Nickisch, 2010) offers a library of functions that are directed towards Gaussian processes for regression and classification in a machine learning context. Finally, recent versions of MATLAB (starting from R2015b) provide a rapidly growing Gaussian process library for regression and classification.

Due to the variety of potential applications of Kriging, different toolboxes tend to be focused on a specific user niche. There is limited availability

of general purpose Kriging toolboxes that allow for seamless integration within various UQ workflows ranging from *e.g.* basic uncertainty propagation to reliability analysis and surrogate-model-based optimisation. To this end, the Kriging toolbox presented here was developed as a module of the general purpose UQ framework, UQLab (Marelli and Sudret (2014), `www.uqlab.com`). In addition, although most of the aforementioned toolboxes offer a significant set of configuration options, the support for fully customisable Kriging is often limited or not easily accessible, which can be a drawback in a research environment. Finally, the user experience may vary from user-friendly to complex (especially to access the most advanced features), often requiring a significant degree of programming knowledge. This might be rather inconvenient for applied scientists and practitioners with limited programming knowledge. The Gaussian process modelling module (GP-module) in UQLab was developed with user-friendliness and customasibility as its core features. This is showcased in the selected application examples in Section B.3.

B.2 THE UQLAB GAUSSIAN PROCESS MODELLING MODULE

B.2.1 *The UQLab project*

UQLab is a software framework developed by the Chair of Risk, Safety and Uncertainty Quantification at ETH Zürich (Marelli and Sudret, 2014). The goal of this project is to provide an uncertainty quantification tool that is accessible also to a non-highly-IT trained scientific audience. Due to the broadness of the UQ scope, a correspondingly general theoretical framework is required. The theoretical backbone of the UQLab software lies in the global uncertainty framework developed by Sudret (2007); De Rocquigny et al. (2008), sketched in Figure B.1a. According to this framework, the solution of any UQ problem can generally be decomposed into the following steps:

Step A Define the *physical model* and the quantities of interest for the analysis. It is a deterministic representation of an arbitrarily complex physical model, *e.g.* a finite element model in civil and mechanical engineering. In this category also lie metamodels, such as Kriging, since once they are calculated they can be used as surrogates of the underlying "true" model.

Step B Identify and quantify the sources of uncertainty in the parameters of the system that serve as input for Step A. They are represented by a set of random variables and their joint probability density function (PDF).

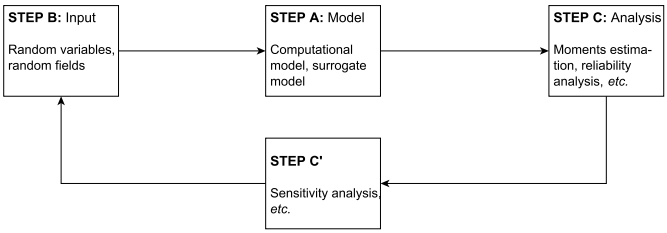
Step C Propagate the uncertainties identified in Step B through the computational model in Step A to characterise the uncertainty in the model response. This type of analyses include moments analysis, full PDF characterisation, rare events estimation, sensitivity analysis, etc.

Step C' Optionally, exploit the by-products of the analysis in Step C to update the sources of uncertainty, *e.g.* by performing model reduction based on sensitivity analysis.

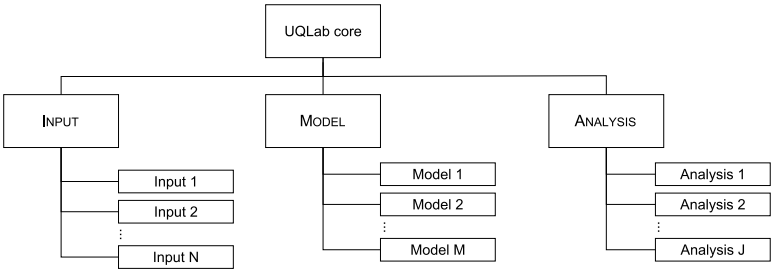
These components introduce a clear semantic distinction between the elements involved in any UQ problem: MODEL, INPUT and ANALYSIS. This theoretical framework provides the ideal foundation for the development of the information flow model in a multi-purpose UQ software.

At the core of UQLAB lies a modular infrastructure that closely follows the semantics previously described, graphically represented in Figure B.1b. The three steps identified in Figure B.1a are directly mapped to *core modules* in Figure B.1b: MODEL corresponds to Step A (physical modelling, metamodeling), INPUT to Step B (sources of uncertainty) and ANALYSIS to Step C (uncertainty analysis). Within the UQLAB framework, a *module* refers to some particular functionality, *e.g.*the GP-module provides Kriging surrogate modelling. Each module extends the functionalities of one of the core modules. It can be either self-contained or capitalise on other modules for extended functionalities.

The real "actors" of a UQ problem are contained in the *objects* connected to each of the core modules. A typical example of such objects would be an INPUT object that generates samples distributed according to arbitrary



(a) The theoretical UQ framework based on which any UQ problem can be described.



(b) The modular structure of the UQLAB framework. An arbitrary number of objects (Input, Model, Analysis) can be connected at any stage of the UQ problem.

FIGURE B.1: An abstract illustration of the UQLAB architecture (b) based on the theoretical UQ framework in (a) by Sudret (2007).

PDFs, a MODEL object that runs a complex FEM simulation, or an ANALYSIS object that performs reliability analysis. The platform allows one to define an arbitrary number of objects and select the desired ones at various stages of the solution of a complex UQ problem.

UQLAB first became freely available to the academic community on July 2015 as a beta version. On April 2017 the version 1.0 of UQLAB was released. Starting from version 1.0 all the scientific code of the software is open-source (BSD license). By September 2017 around 1000 users have already registered and used it.

B.2.2 *The GP-module*

Kriging is one of the metamodelling modules available in UQLAB (Lataniotis et al., 2018, 2019). Following the semantics described in the previous section,

it is attached to the MODEL core module. Although the GP-module itself can be used by other modules, *e.g.*an ANALYSIS module performing reliability analysis combining Kriging and Monte Carlo Simulation (AK-MCS) (Echard et al., 2011; Marelli et al., 2019), the focus of this work is on the capabilities of the GP-module itself.

An overview of the available features of the GP-module is given in Table B.1. The GP-module incorporates the four ingredients identified in Section 4.2:

- *Trends*: Universal Kriging trends are fully supported, including simple, ordinary, or polynomial of arbitrary degree. In addition, custom basis functions $f(\mathbf{x})$ or a completely custom trend function may be specified
- *Correlation functions*: Standard correlation families from the literature are readily available as well as the possibility of creating user-defined ones. For multi-dimensional inputs ellipsoidal and separable correlation functions can be used, allowing also for isotropic ones. Fully user-specified correlation functions are also supported
- *Estimation methods*: Maximum likelihood (Eq. (4.22)) and cross-validation (Eq. (4.24)) methods can be used for estimating the hyperparameters
- *Optimisation methods*: MATLAB's built-in local and global optimisation methods are offered, namely BFGS and genetic algorithm as well as genetic algorithm with BFGS refinement (hybrid).

In addition, various scaling operations are allowed for avoiding numerical instabilities during the hyperparameters estimation. Such operations may vary from simple zero-mean scaling to more advanced ones such as isoprobabilistic transformations by interfacing with other UQLAB modules.

Following the general design principle of UQLAB concerning user-friendliness, all the possible configuration options have default values pre-assigned to allow basic usage of the module with very few lines of code (see Section B.3.1). A MATLAB structure variable is used to specify a Kriging configuration, called `KOptions` in the following sections.

To showcase the minimal working code for obtaining a Kriging surrogate a simple application is considered. The experimental design consists of 8

random samples in the $[0, 15]$ interval and it is contained in the variable `XED`. The "true" model is $\mathcal{M}(x) = x\sin(x)$ and the corresponding model responses are stored in the variable `YED`. The minimal code required for obtaining a Kriging surrogate, given `XED` and `YED` is the following:

```
KOptions.Type = 'Metamodel';
KOptions.MetaType = 'Kriging';
KOptions.ExpDesign.X = XED;
KOptions.ExpDesign.Y = YED;
myKriging = uq_createModel(KOptions);
```

The first line clarifies the type of UQLab object that is being requested. Following the general UQ Framework in Figure B.1a a MODEL object of type `'Metamodel'` is created. The next line specifies the type of metamodel, followed by the manual specification of the experimental design. Finally the UQLab command `uq_createModel` is used in order to create a MODEL object using the configuration options in `KOptions`.

The resulting Kriging metamodel object `myKriging` contains all the required information to compute the mean and variance of the Kriging predictor on new test points (X). This can be done using the following command:

```
[meanY, varY] = uq_evalModel(myKriging, X);
```

where `meanY` corresponds to the mean and `varY` to the variance of the Kriging predictor on the test points (see Eqs. (4.8), (4.9)).

Once the metamodel is created, a report of the main properties of the Kriging surrogate model can be printed on screen by:

```
uq_print(myKriging);
```

```
%-------------- Kriging metamodel --------------%
Object Name:       Model 1
Input Dimension:   1

Experimental Design
  Sampling:        User
  X size:          [8x1]
  Y size:          [8x1]
```

TABLE B.1: List of features of the UQLAB GP-module. The default values for each property is in bold.

Feature	Specification	Value	Description
Trend		Simple	A constant term specified by the user (simple Kriging)
		Ordinary	A constant term estimated using Eq. (4.10) (ordinary Kriging)
		Polynomial basis	The trend in Eq. (4.15) consists of polynomial basis functions f_k of arbitrary degree
		Custom basis	The trend in Eq. (4.15) consists of arbitrary functions f_k
		Custom trend	Custom trend function that computes $\boldsymbol{F}$ directly
Correlation	Types	Separable	As described in Eq. (4.16). Both isotropic and anisotropic variants are supported.
		Ellipsoidal	As described in Eq. (4.17). Both isotropic and anisotropic variants are supported.
		Custom	Custom correlation function that computes $\boldsymbol{R}$ directly
	Families	Commonly used	All the correlation families reported in Table 4.2 are available
		Custom	A custom correlation family can be specified
Estimation		ML	Maximum-likelihood estimation (see Eq. (4.22))
		CV	K-fold Cross-Validation method (see Eq. (4.24)). Any K value is supported
Optimisation		BFGS	Gradient-based optimisation method (Broyden-Fletcher-Goldfarb-Shanno algorithm). MATLAB built-in
		GA	Global optimisation method (genetic algorithm). MATLAB built-in
		HGA	Genetic algorithm optimisation with BFGS refinement

```
Trend
  Type:             ordinary
  Degree:           0

Gaussian Process
  Corr. Type:       ellipsoidal(anisotropic)
  Corr. family:     matern-5_2
  sigma^2:          4.787983e+01
Estimation method:  Cross-Validation

Hyperparameters
  theta:             [ 0.00100 ]
Optim. method:       Hybrid Genetic Algorithm

Leave-one-out error: 4.3698313e-01
%------------------------------------------------%
```

It can be observed that the default values for the trend, correlation function, estimation and optimisation method have been assigned (see Table B.1). A visual representation of the metamodel can be obtained by:

```
uq_display(myKriging);
```

Note that the `uq_display` command can only be used for quickly visualising Kriging surrogates when the inputs are one- or two-dimensional. The figure produced by `uq_display` is shown in Figure B.2.

B.3 APPLICATION EXAMPLES

B.3.1 *Basic example*

The goal of this introductory example is to calculate a Kriging surrogate of the Branin-Hoo function based on a limited set of observations. The Branin-Hoo function reads (Forrester et al., 2008):

$$\mathcal{M}(\mathbf{x}) = a\left(x_2 - bx_1^2 + cx_1 - r^2\right)^2 + s\,(1-t)\cos(x_1) + s\,,\ \mathbf{x} \in \mathbb{R}^2. \quad \text{(B.1)}$$

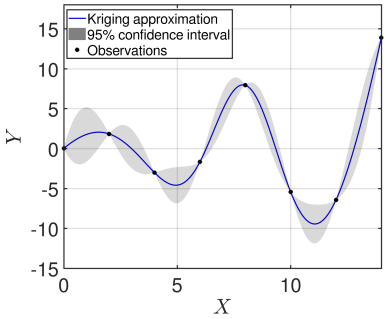


FIGURE B.2: The output of `uq_display` of a Kriging MODEL object having a one-dimensional input.

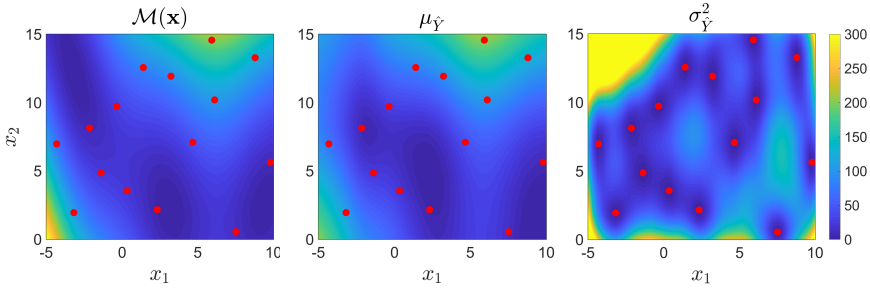


FIGURE B.3: From left to right: the Branin-Hoo function (true model) followed by the mean and variance of the Kriging predictor. The experimental design is illustrated by red dots

Some standard values of the parameters are used, namely $a = 1$, $b = 5.1(4\pi^2)$, $c = 5/\pi$, $r = 6$, $s = 10$ and $t = 1/(8\pi)$. The function is evaluated on the square $x_1 \in [-5, 10]$, $x_2 \in [0, 15]$.

By taking advantage of the INPUT and MODEL modules of UQLAB, the experimental design and model responses that will be used for calculating the surrogate can be generated with minimal effort. First, the probabilistic input model and the true model are defined as follows:

```
% Start the UQLab framework
uqlab;
% Specify the probabilistic input model
```

```
IOptions.Marginals(1).Type = 'Uniform';
IOptions.Marginals(1).Parameters = [-5, 10];
IOptions.Marginals(2).Type = 'Uniform';
IOptions.Marginals(2).Parameters = [0, 15];
myInput = uq_createInput(IOptions);
% Specify the computational model
MOptions.mString = ['(X(:,2) - 5.1/(2*pi)^2*X(:,1).^2 ...
+ 5/pi*X(:,1) - 6).^2' '+ 10*(1-1/(8*pi))*cos(X(:,1)) + 10'];
myModel = uq_createModel(MOptions);
```

Note that the MODEL object of the Branin-Hoo function can be equally coded in a MATLAB m-file or written as a string (which is a useful feature for simple demo functions only).

Next, the experimental design `XED` is generated along with the corresponding true model responses `YED`. The Latin Hypercube Sampling (LHS) method is used to obtain a space-filling experimental design of 15 samples (McKay et al., 1979):

```
% Draw 15 samples using Latin Hypercube Sampling
XED = uq_getSample(15, 'LHS');
% Calculate the corresponding model responses
YED = uq_evalModel(myModel, XED);
```

A Kriging surrogate model using the `XED`, `YED` variables can be created as follows:

```
KOptions.Type = 'Metamodel';
KOptions.MetaType = 'Kriging';
KOptions.ExpDesign.Sampling = 'user';
KOptions.ExpDesign.X = XED;
KOptions.ExpDesign.Y = YED;
myKriging = uq_createModel(KOptions);
```

All the required ingredients for obtaining a Kriging surrogate are assigned default values unless specified by the user (see Section B.2.2). The surrogate that is obtained can be visually inspected by issuing the command:

```
uq_display(myKriging);
```

The result of the `uq_display` command is shown in Figure B.3. The Kriging surrogate `myKriging` can be used like any other model (*e.g.* `myModel`) to calculate its response given a new sample of the input `X` using the `uq_evalModel` function. For example, the mean predictor, `meanY`, of 100 samples generated by Monte Carlo sampling can be computed as follows:

```
X = uq_getSample(100);
meanY = uq_evalModel(myKriging, X);
```

More information can be extracted from the Kriging predictor using a slightly different syntax. The following code:

```
[meanY, varY, covY] = uq_evalModel(myKriging, X);
```

allows to retrieve the 100×1 Kriging mean `meanY`, the 100×1 Kriging variance `varY` and the 100×100 full covariance matrix of the surrogate model responses `covY`.

B.3.2 *Hierarchical Kriging*

To further illustrate the flexibility that can be achieved with the use of arbitrary trend functions, a hierarchical Kriging application is showcased. Hierarchical Kriging (Han et al., 2012) is one Kriging extension aiming to fuse information from experimental designs related to different physical models of different fidelity. This is achieved by first calculating a Kriging surrogate using the low-fidelity observations and then using it as the trend of the high-fidelity surrogate. This approach can be extended to more fidelity levels in a similar fashion. A set of observations and model responses is used that originates from aero-servo-elastic simulations of a wind-turbine as presented in Abdallah et al. (2019). Given a set of input parameters related to the wind flow, the output of interest is the maximal bending moment at the blade root of a wind turbine.

Two types of simulators are available for estimating the maximal bending moment given the wind conditions. A low-fidelity simulator can generate estimates of the output with minimal computation time at the cost of lower accuracy. On the other hand a high-fidelity simulator can more accurately predict the maximal bending moment at a significantly higher computational cost. In this example a total of 300 low-fidelity and 15 high-fidelity simulations are available. First a Kriging surrogate is computed on

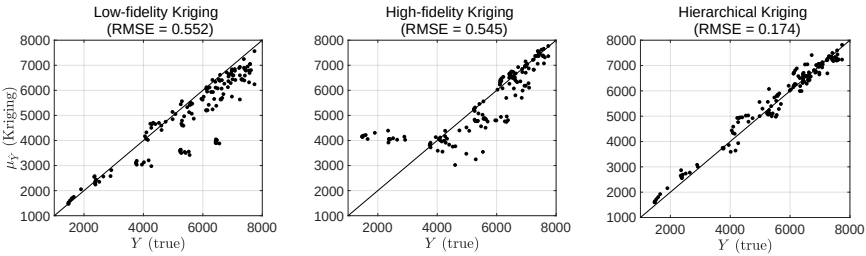


FIGURE B.4: Comparison of true model output (from high fidelity simulations) versus various Kriging surrogates on a validation set of size 150.

the low-fidelity dataset that is contained in variables `XED_LF, YED_LF` as follows:

```
% Create the low-fidelity surrogate
KOptions_LF.Type = 'Metamodel';
KOptions_LF.MetaType = 'Kriging';
KOptions_LF.ExpDesign.X = XED_LF;
KOptions_LF.ExpDesign.Y = YED_LF;
KOptions_LF.Corr.Family = 'Matern-3_2';

myKriging_LF = uq_createModel(KOptions_LF);
```

Using the same configuration options, another Kriging surrogate is computed using the high-fidelity dataset (`XED_HF` and `YED_HF`):

```
% Create the high-fidelity surrogate
KOptions_HF.Type = 'Metamodel';
KOptions_HF.MetaType = 'Kriging';
KOptions_HF.ExpDesign.X = XED_HF;
KOptions_HF.ExpDesign.Y = YED_HF;
KOptions_HF.Corr.Family = 'Matern-3_2';

myKriging_HF = uq_createModel(KOptions_HF);
```

Now a hierarchical Kriging surrogate is computed which is trained on the high-fidelity dataset but uses the low-fidelity Kriging surrogate (*i.e.*its mean predictor) as trend:

```
% Create the hierarchical Kriging surrogate
KOptions_Hier.Type = 'Metamodel';
KOptions_Hier.MetaType = 'Kriging';
KOptions_Hier.ExpDesign.X = XED_HF;
KOptions_Hier.ExpDesign.Y = YED_HF;
KOptions_Hier.Corr.Family = 'Matern-3_2';
KOptions_Hier.Trend.Type = 'custom';
KOptions_Hier.Trend.CustomF = @(x) uq_evalModel(myKriging_LF, x);
KOptions_Hier.Scaling = false;

myKriging_Hier = uq_createModel(KOptions_Hier);
```

The option `KOptions_Hier.Scaling` refers to the scaling of the input space before computing the surrogate model. In case of hierarchical Kriging scaling should be disabled because the low-fidelity surrogate is calculated on the original data and needs to be used "as is".

The performance of the different surrogate models is tested on a separate validation set of 150 high-fidelity simulations that is contained in the variables `XVAL_HF` and `YVAL_HF`. The output mean Kriging predictor on the validation set is calculated as follows:

```
meanY_LF = uq_evalModel(myKriging_LF, XVAL_HF);
meanY_HF = uq_evalModel(myKriging_HF, XVAL_HF);
meanY_Hier = uq_evalModel(myKriging_Hier, XVAL_HF);
```

where `meanY_LF`, `meanY_HF` and `meanY_Hier` correspond to the low-fidelity, high-fidelity and hierarchical Kriging predictors respectively.

In Figure B.4 a comparison of the true model output `YVAL_HF` versus the mean Kriging predictors is made. In each case the Root Mean Square Error (RMSE) is reported for quantifying the predictive performance of the surrogate:

$$E_{RMSE} = \frac{1}{N\mathrm{Var}\left[Y\right]} \sum_{i=1}^{N} \left(Y^{(i)} - \mu_{\hat{Y}}^{(i)} \right)^2 \tag{B.2}$$

where Y denotes the true model outputs (in this case `YVAL_HF`), $\mu_{\hat{Y}}$ the Kriging predictor mean (in this case variables `meanY_LF`, `meanY_HF` and

`meanY_Hier` for each surrogate, respectively) and N the number of samples in the validation set.

In this example, by taking advantage of the low-cost, low-fidelity observations, the hierarchical Kriging predictor achieves a 68% decrease of the RMSE on the validation set compared to the Kriging model that was solely based on the high-fidelity measurements. Moreover, by inspecting the mean responses of each Kriging predictor in Figure B.4 it is clear that the hierarchical Kriging surrogate significantly reduces the prediction bias compared to the low- and high-fidelity ones taken as standalone. As demonstrated by this application, building a hierarchical Kriging surrogate model requires minimal effort thanks to the customisability of the GP-module.

B.3.3 *Kriging with custom correlation function*

This example illustrates how the correlation function customisation capabilities of the GP-module can be used to apply Kriging in a non-standard setting.

Consider the discontinuous subsurface model given in Figure B.5, which may represent the distribution of some soil property (e.g. porosity) in the presence of a fault. The true model consists in two realisations of two distinct random processes on the two regions A_1 and A_2 at the left and right of the fault, respectively:

$$\mathcal{M}(\mathbf{x}) = \begin{cases} Z_1(\mathbf{x}, R(\boldsymbol{\theta}_1)), & \mathbf{x} \in A_1 \\ Z_2(\mathbf{x}, R(\boldsymbol{\theta}_2)), & \mathbf{x} \in A_2 \end{cases} \tag{B.3}$$

where $\boldsymbol{x} = \{x_1, x_2\}$ represents the spatial coordinates in the 2D domain, Z_1 (resp. Z_2) are realisations of a Gaussian process characterised by a correlation function with length scales $\boldsymbol{\theta}_1 = \{\theta_{11}, \theta_{12}\}$ (resp. $\boldsymbol{\theta}_2 = \{\theta_{21}, \theta_{22}\}$).

A Kriging surrogate model will be calculated using the following correlation function:

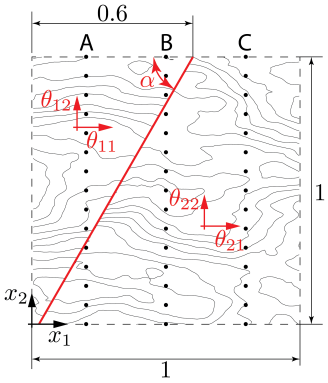


FIGURE B.5: Graphical visualisation of the subsurface model. The unknowns (length scales of each random field and the fault angle) are denoted by red colour.

$$R(\mathbf{x},\mathbf{x}';\boldsymbol{\theta}) = \begin{cases} R(\mathbf{x},\mathbf{x}';\widehat{\boldsymbol{\theta}}_1), & (\mathbf{x},\mathbf{x}') \in A_1 \times A_1 \\ R(\mathbf{x},\mathbf{x}';\widehat{\boldsymbol{\theta}}_2), & (\mathbf{x},\mathbf{x}') \in A_2 \times A_2 \\ 0 & \text{otherwise} \end{cases} \tag{B.4}$$

where $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \alpha\}$. There is a smooth dependence on x_1, x_2 within each region, but no correlation between points that belong to different regions. The boundary between the two regions is fully defined by the crack angle, α, which is unknown and the fault location that is assumed to be known ($\{x_1, x_2\} = \{0.6, 1\}$). The goal here is to use Kriging to interpolate the measurements taken at borehole locations A,B and C and estimate the 5 unknown parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \alpha\}$. The correlation function of each region is the same, both in the true model and the Kriging surrogate, *i.e.*it is assumed to be known. In particular, the correlation function is separable Matérn 3/2 (see Eq. (4.16) and Table 4.2). The maximum-likelihood method is selected for estimating $\boldsymbol{\theta}$. Due to the complexity of the underlying optimisation problem a hybrid genetic algorithm with a relatively large population size and maximum number of generations is selected.

A MATLAB implementation of the correlation function in Eq. (B.4) is given in Appendix B.5. This MATLAB function is called `my_eval_R` in the following code snippet.

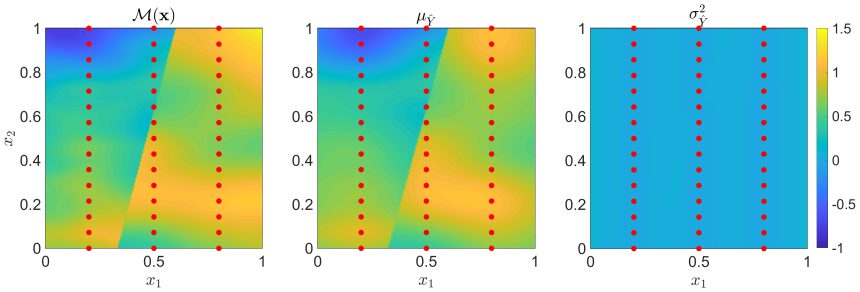


FIGURE B.6: From left to right: The true permeability of the soil, followed by the mean and variance of the Kriging predictor. The experimental design is illustrated by red dots.

The Kriging surrogate is created next, based on a limited set of observations contained in the variables `BoreholeLocations` and `BoreValues`, which contain the locations of the measurements along the boreholes and the value of the desired property, respectively.

```
KOptions.Type = 'Metamodel';
KOptions.MetaType = 'Kriging';
KOptions.ExpDesign.X = BoreholeLocations;
KOptions.ExpDesign.Y = BoreValues;
KOptions.Corr.Handle = @my_eval_R;
% Add upper and lower bounds on the optimization variables
BoundsL = [0.3 0.1 0.3 0.1 pi/6] ;
BoundsU = [0.9 0.5 0.9 0.5 5*pi/6] ;
KOptions.Optim.Bounds =[BoundsL ;BoundsU];
KOptions.Optim.Method = 'HGA';
KOptions.Optim.HGA.nPop = 60;
KOptions.Optim.MaxIter = 50;
KOptions.EstimMethod = 'ML';
KOptions.Scaling = False;

myKriging = uq_createModel(KOptions);
```

Once the Kriging metamodel has been computed, the mean and variance of the Kriging predictor can be quickly visualised for 1D and 2D models using the `uq_display` command, which produces a plot similar to Figure B.6, except in a smaller domain determined by the range of the points in the ex-

Parameter	θ_{11}	θ_{12}	θ_{21}	θ_{22}	α
True value	0.600	0.250	0.900	0.350	1.309
Estimated value	0.310	0.271	0.310	0.374	1.342
Relative error (%)	48.3	8.2	65.6	6.9	2.5

TABLE B.2: Listing of the true and estimated correlation function parameters, $\boldsymbol{\theta}$, for the Kriging surrogate of the subsurface model.

perimental design. A comparison between the true and the estimated values of $\boldsymbol{\theta}$ is given in Table B.2. As expected, the accuracy of the hyperparameters estimation is low due to the limited dispersion of the experimental design. The error of the length scale estimates along the x_1 direction is consistently larger due to the lack of samples along that direction. From a coding perspective, although the correlation function that is used is relatively complex, it is straightforward to use in a Kriging surrogate once coded as a MATLAB function (by setting the `KOptions.Corr.Handle` value appropriately). Moreover, custom correlation functions are allowed to have an arbitrary number of hyperparameters. The only requirement is that the optimisation bounds (or initial value, depending on the optimisation method that is used) must have the same length as the number of the hyperparameters.

B.4 SUMMARY AND OUTLOOK

In this paper the GP-module of the UQLAB software framework was presented. This UQLAB module enables practitioners from various disciplines to get started with Kriging metamodelling with minimal effort as was illustrated in the introductory application in Section B.3.1. However, it is also possible to access more advanced customisation, *e.g.* for research purposes. This was showcased in Section B.3.2 where a hierarchical Kriging metamodel was developed and in Section B.3.3 where a relatively complex, non-stationary correlation function was used to solve a geostatistical inverse problem. The GP-module is freely available to the academic community since the first beta release of UQLAB in July 2015.

Future improvements of the GP-module include offering support for noisy-observations (*a.k.a.* Gaussian process regression) and providing built-in optimisation tools to relax the current toolbox requirements of the GP-module.

In addition to the modules currently exploiting its functionality (Polynomial Chaos-Kriging and Reliability analysis (Marelli et al., 2019; Schöbi et al., 2019)), new UQLab modules that interface with the GP-module are currently under active development. The upcoming random fields module will offer several random field types (conditional and unconditional) together with advanced sampling methodologies and will be interfaced with the GP-module to offer trajectory resampling capabilities. The upcoming Reliability-Based Design Optimisation (RBDO) module uses the surrogate modelling capabilities of the GP-module for solving RBDO problems as described in Moustapha et al. (2016).

B.5 KRIGING WITH CUSTOM CORRELATION FUNCTION: IMPLEMENTATION DETAILS

The aim of this section is to provide some additional implementation details on the application example in Section B.3.3, in terms of the MATLAB code involved. The correlation function described in Eq. (B.4) can be translated to the following MATLAB function:

```
function R = my_eval_R( x1,x2,theta,parameters )

xc = 0.6; % the x-location of the crack on the surface
yc = 1  ; % the y-location of the crack on the surface

length_scales_1 = theta(1:2);
length_scales_2 = theta(3:4);
crack_angle     = theta(5)  ;

% find the angles of each sample of x1
angles_x1 = acos(  (xc - x1(:,1))./sqrt((x1(:,1) - xc).^2 + ...
(x1(:,2) - yc).^2 ) );
% find the indices of x1 that belong to first region
idx_x1_1 = angles_x1 <= crack_angle;
% find the indices of x1 that belong to second region
idx_x1_2 = ~idx_x1_1;
% find the angles of each sample of x2
angles_x2 = acos(  (xc - x2(:,1))./sqrt((x2(:,1) - xc).^2 + ...
(x2(:,2) - yc).^2 ) );
% find the indices of x2 that belong to first region
```

```
idx_x2_1 = angles_x2 <= crack_angle;
% find the indices of x2 that belong to second region
idx_x2_2 = ~idx_x2_1;

% set-up various correlation function options so that we can
% re-use the built-in UQLab function for evaluating R in each
% region
CorrOptions.Type = 'separable';
CorrOptions.Family = 'Matern-3_2';
CorrOptions.Isotropic = false;
CorrOptions.Nugget = 1e-2;

% initialize R matrix
R = zeros(size(x1,1), size(x2,1));

% Compute the R values in region 1
R(idx_x1_1,idx_x2_1) = uq_Kriging_eval_R( ...
x1(idx_x1_1,:), x2(idx_x2_1,:), length_scales_1, CorrOptions);
% Compute the R values in region 2
R(idx_x1_2,idx_x2_2) = uq_Kriging_eval_R( ...
x1(idx_x1_2,:), x2(idx_x2_2,:), length_scales_2, CorrOptions);

end
```

The provided code, although vectorised, is optimised for readability and not performance. To that end, the internal function of the GP-module `uq_Kriging_eval_R` is used for calculating the correlation function value in each of the regions.

C

EXTENDED RESULTS AND IMPLEMENTATION REMARKS

C.1 IMPLEMENTATION DETAILS REGARDING THE DRSM APPLICATIONS

This section provides an extensive list of the configuration parameter values that were used to produce the results in Section 5.4. Table C.1 lists the configuration parameters of Kriging surrogate models whereas Table C.2 of PCE ones. For each surrogate method a distinction is made, in terms of the parameters used, between the proxy (*i.e.* low computational cost) surrogate and the high-accuracy one. The proxy surrogates were used for solving the nested optimisation problem of DRSM in Eqs. (5.6), (5.7). The same configuration was used to calculate the high-accuracy surrogates regardless of the input compression method (DRSM or disjoint PCA/KPCA).

The parameters of the DRSM-based optimisation are listed in Table C.3. Note that, the exact same optimisation algorithm and parameters were used for optimising $\mathbf{w}$ w.r.t. the KPCA reconstruction and point-wise distance error in the method comparison box-plots. Note that the optimisation constraints differ from the ones reported in Table C.3 when a polynomial kernel is used in KPCA (see Table 3.1) for improved numerical stability of the solver. On top of the box constraints reported in the table, that still apply for w_1 and w_2, the variable w_3 (degree) is constrained to integer values $1 \leq w_3 \leq 4$ instead. In addition, the following non-linear constraint is included:

$$w_1 \mathbf{x}^\mathsf{T} \mathbf{x}' + w_2 > 1. \tag{C.1}$$

TABLE C.1: The configuration of the Kriging surrogates that were calculated during the various steps of DRSM for each application example.

Application	Sobol' function	Resistor networks	2D diffusion
1. Proxy surrogate configuration			
Trend	constant ($P = 0$)	linear ($P = 1$)	linear ($P = 1$)
Correlation family	*isotropic* Matèrn (Table 4.2) with $\nu = 5/2$		
Estimation method	Cross-validation (Eq. (4.24))		
Optim. method	Genetic algorithm (GA) with BFGS (gradient based) refinement of final solution		
Optim. contraints	$\boldsymbol{\theta} \in [0.01, 100]$		
Population size (GA)	10		
Max. iterations:	20 for both GA and BFGS		
2. High-accuracy surrogate configuration. Only the parameters that differ from the proxy surrogate configuration are listed			
Correlation family	*anisotropic* Matèrn with $\nu = 5/2$		
Population size (GA)	20		
Max. iterations:	50 for both GA and BFGS		

TABLE C.2: The configuration of the PCE surrogates that were calculated during the various steps of DRSM for each application example

Application	Sobol' function	Resistor networks	2D diffusion
1. Proxy surrogate configuration			
Coeff. calculation method	Ordinary least squares (Berveiller et al., 2006)		
Univariate polynomials family	Legendre		
Hyperbolic truncation q (Blatman and Sudret, 2010)	0.75	0.50	0.65
Polynomial degree (adaptive search range)	$[1, 10]$	$[1, 10]$	$[1, 5]$
2. High-accuracy surrogate configuration. Only the parameters that differ from the proxy surrogate configuration are listed			
Coeff. calculation method	Hybrid least angle regression (Blatman and Sudret, 2011)		
Univariate polynomials family	Orthogonal to the probability density function of the input variables that is estimated by kernel-smoothing, using the Stieltjes procedure (Gautschi, 2004)		
Hyperbolic truncation q (Blatman and Sudret, 2010)	0.75		
Polynomial degree (adaptive search range)	$[1, 15]$		

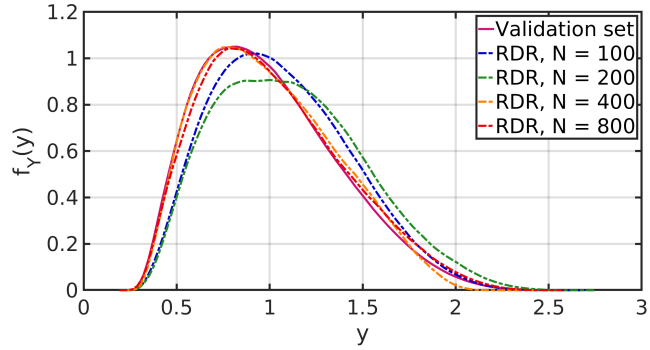
TABLE C.3: Parameters of the DRSM optimisation algorithm

Application	Sobol' function	Resistor networks	2D diffusion
Optim. method	Genetic algorithm with BFGS (gradient based) refinement of final solution		
Optim. constraints	$\mathbf{w} \in [0.1, 300]$		
Population size(GA):	20 for isotropic KPCA kernels, 80 for anisotropic	20 for isotropic KPCA kernels, 100 for anisotropic	20 (only isotropic KPCA kernels were considered)
Max. iterations:	80 for both GA and BFGS	150 for both GA and BFGS	80 for both GA and BFGS

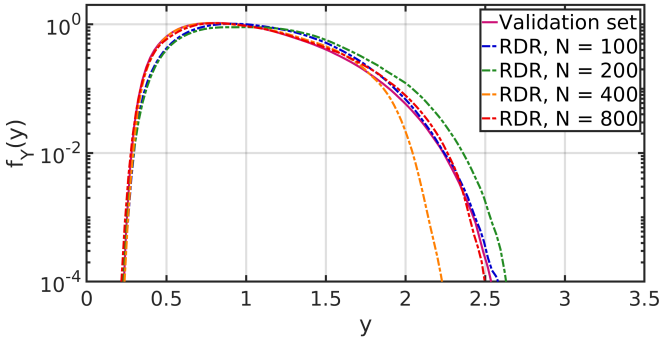
C.2 EXTENDED RESULTS ON HIGH-DIMENSIONAL UNCERTAINTY PROPAGATION

In the following sections we provide additional visualisations related to the performance of the reduced dimension resampling technique, applied on the benchmark applications in Section 6.3. In Section C.2.1 (resp. Sections C.2.2 and C.2.3) two groups of plots are shown. The first group provides a comparison between the true f_Y (inferred from the validation set) and the approximate PDFs that where calculated by the RDR technique using a varying number of experimental design samples $N \in \{100, 200, 400, 800\}$. The second group of plots shows an extended version of the quantile estimates by RDR discussed in Section 6.3.1 (resp. Sections 6.3.2 and 6.3.3), including the N cases that were omitted from the main text.

C.2.1 *Sobol function*

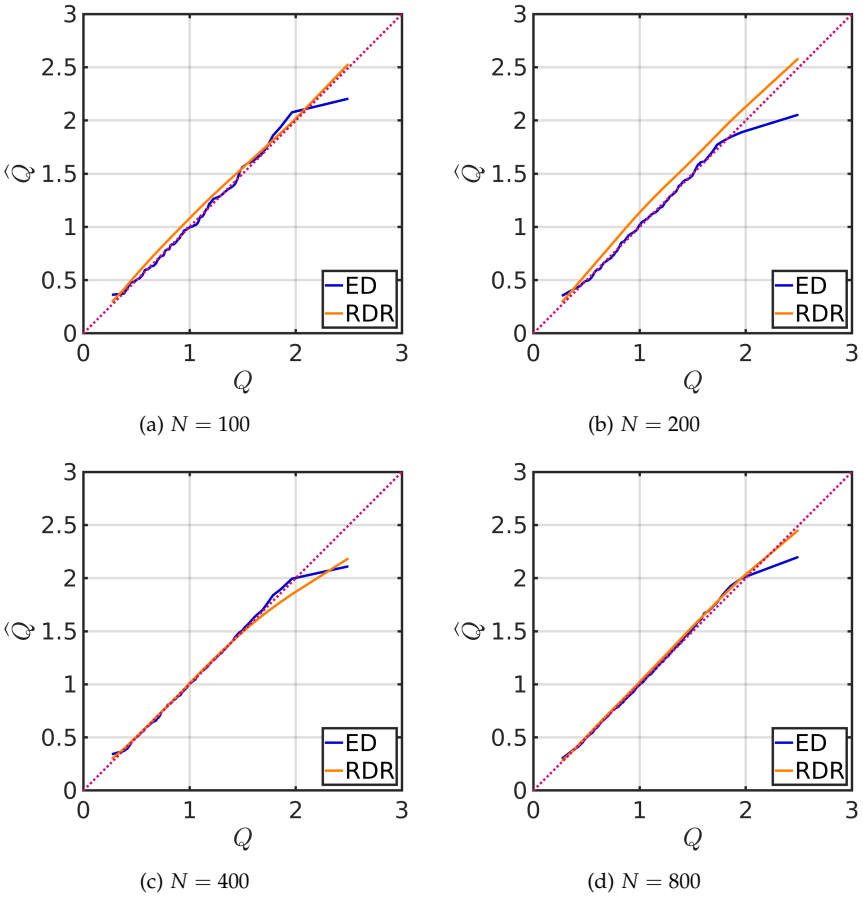


(a) Vertical axis in linear scale



(b) Vertical axis in logarithmic scale

FIGURE C.1: Sobol function: comparison between the response PDF approximation using either true or artificial samples by RDR, using an experimental design of varying size.

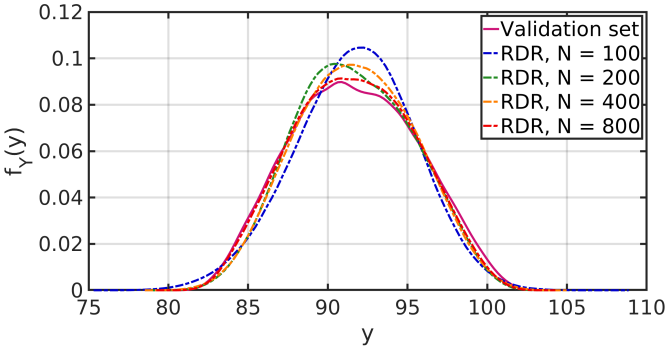


(a) $N = 100$ (b) $N = 200$

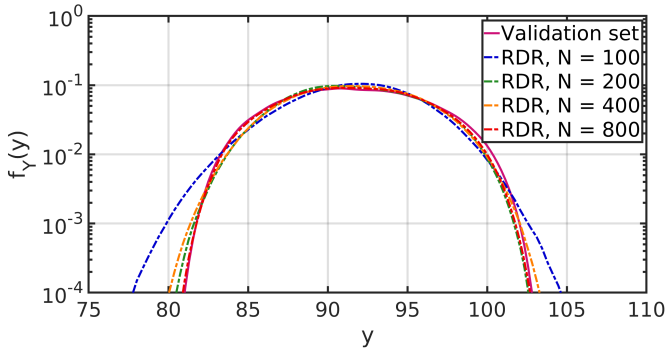
(c) $N = 400$ (d) $N = 800$

FIGURE C.2: Sobol function: quantile estimates comparison.

C.2.2 *Electrical resistor network*

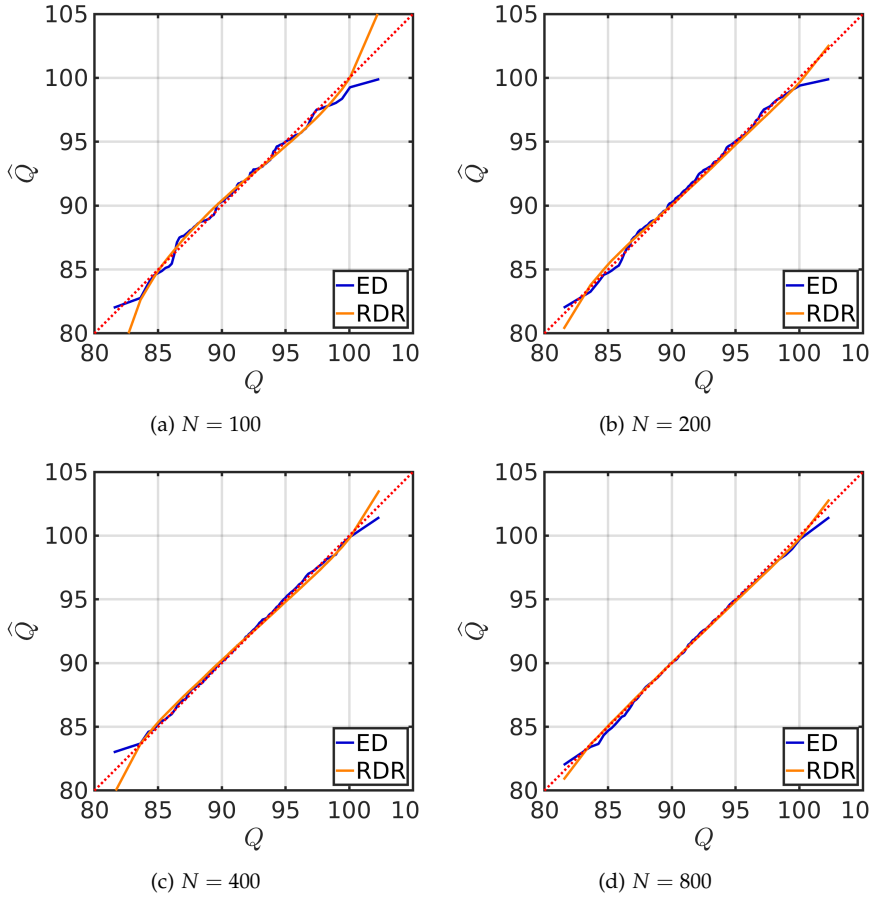


(a) Estimated PDF comparison



(b) Estimated PDF comparison (logarithmic scale)

FIGURE C.3: Resistor networks: comparison between the response PDF approximation using either true or artificial samples by RDR, using an experimental design of varying size.

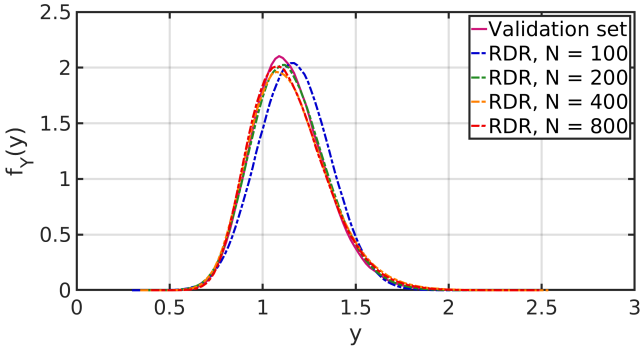


(a) $N = 100$ (b) $N = 200$

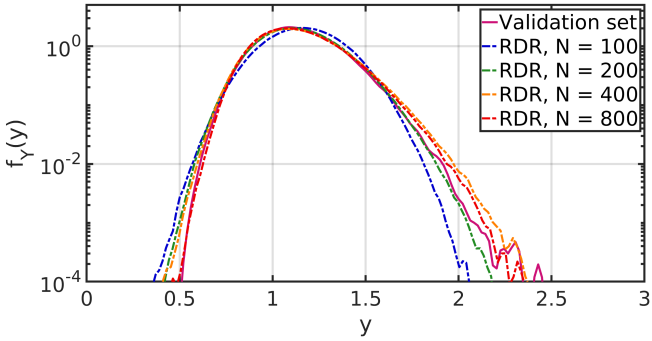
(c) $N = 400$ (d) $N = 800$

FIGURE C.4: Resistor networks: quantile estimates comparison.

C.2.3 *2D heat diffusion*

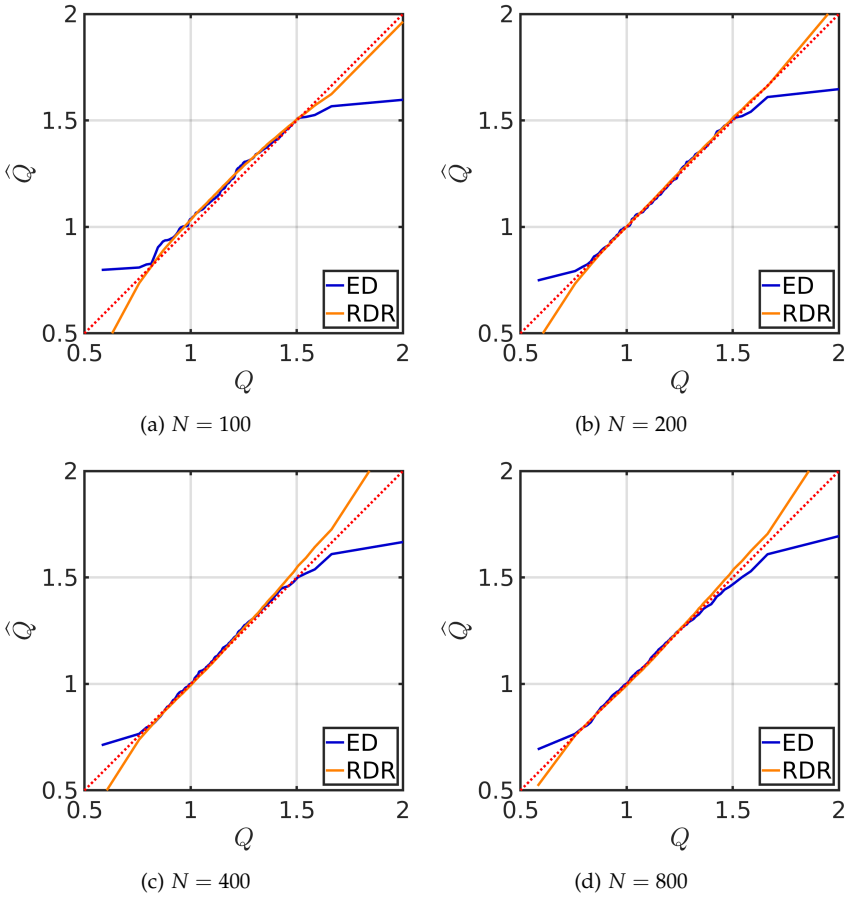


(a) Estimated PDF comparison



(b) Estimated PDF comparison (logarithmic scale)

FIGURE C.5: 2D heat diffusion: comparison between the response PDF approximation using either true or artificial samples by RDR, using an experimental design of varying size.



(a) $N = 100$ (b) $N = 200$

(c) $N = 400$ (d) $N = 800$

FIGURE C.6: 2D heat diffusion: quantile estimates comparison.

C.3 TIME SERIES FEATURE EXTRACTION USING TSFRESH

For the SHM application in Chapter 7, the PYTHON package TSFRESH was used in some of the compression setups, as shown in Figure 7.3. It corresponds to a transform of the form $g_{TSFRESH}(\mathbf{X}) = \{A_1(\mathbf{X}), \ldots, A_{M_A}(\mathbf{X})\}$, where each feature $A_i(\mathbf{X})$ corresponds to a different operation on $\mathbf{X}$. Notice that feature extraction may not necessarily imply dimensionality reduction. The main motivation of this step is not to compress the input space but

to identify an alternative representation that makes it more suitable for constructing a surrogate model.

The list of the extracted features is given in Table C.4. To understand the exact expression of each $A_i(\mathbf{X})$, this list should be used in conjunction to the reference list provided by Christ et al. (2018). We adopt the naming scheme of TSFRESH, where the name of each feature is constructed by concatenating:

- The name of the feature extractor. This refers to one of the available methods in `tsfresh.feature_extraction.feature_calculators`
- The name and value of each parameter supplied to the feature extractor (if any)
- The name of the attribute that the feature corresponds to. This is only relevant for feature extractors that return more than one values (attributes)

After calculating all the features listed below, a simple selection step takes place. Only the ones with non-zero variance across the experimental design are retained. This results to using 698 features for each inflow wind component (as shown in Figure 7.3) instead of 788.

TABLE C.4: The features extracted using TSFRESH.

Index	Name
1	abs_energy
2	absolute_sum_of_changes
3	agg_autocorrelation__f_agg_"mean"
4	agg_autocorrelation__f_agg_"median"
5	agg_autocorrelation__f_agg_"var"
6	agg_linear_trend__f_agg_"max"__chunk_len_10__attr_"intercept"
7	agg_linear_trend__f_agg_"max"__chunk_len_10__attr_"rvalue"
8	agg_linear_trend__f_agg_"max"__chunk_len_10__attr_"slope"
9	agg_linear_trend__f_agg_"max"__chunk_len_10__attr_"stderr"
10	agg_linear_trend__f_agg_"max"__chunk_len_50__attr_"intercept"
11	agg_linear_trend__f_agg_"max"__chunk_len_50__attr_"rvalue"
12	agg_linear_trend__f_agg_"max"__chunk_len_50__attr_"slope"

TABLE C.4: The features extracted using TSFRESH.

Index	Name
13	agg_linear_trend__f_agg_"max"__chunk_len_50__attr_"stderr"
14	agg_linear_trend__f_agg_"max"__chunk_len_5__attr_"intercept"
15	agg_linear_trend__f_agg_"max"__chunk_len_5__attr_"rvalue"
16	agg_linear_trend__f_agg_"max"__chunk_len_5__attr_"slope"
17	agg_linear_trend__f_agg_"max"__chunk_len_5__attr_"stderr"
18	agg_linear_trend__f_agg_"mean"__chunk_len_10__attr_"intercept"
19	agg_linear_trend__f_agg_"mean"__chunk_len_10__attr_"rvalue"
20	agg_linear_trend__f_agg_"mean"__chunk_len_10__attr_"slope"
21	agg_linear_trend__f_agg_"mean"__chunk_len_10__attr_"stderr"
22	agg_linear_trend__f_agg_"mean"__chunk_len_50__attr_"intercept"
23	agg_linear_trend__f_agg_"mean"__chunk_len_50__attr_"rvalue"
24	agg_linear_trend__f_agg_"mean"__chunk_len_50__attr_"slope"
25	agg_linear_trend__f_agg_"mean"__chunk_len_50__attr_"stderr"
26	agg_linear_trend__f_agg_"mean"__chunk_len_5__attr_"intercept"
27	agg_linear_trend__f_agg_"mean"__chunk_len_5__attr_"rvalue"
28	agg_linear_trend__f_agg_"mean"__chunk_len_5__attr_"slope"
29	agg_linear_trend__f_agg_"mean"__chunk_len_5__attr_"stderr"
30	agg_linear_trend__f_agg_"min"__chunk_len_10__attr_"intercept"
31	agg_linear_trend__f_agg_"min"__chunk_len_10__attr_"rvalue"
32	agg_linear_trend__f_agg_"min"__chunk_len_10__attr_"slope"
33	agg_linear_trend__f_agg_"min"__chunk_len_10__attr_"stderr"
34	agg_linear_trend__f_agg_"min"__chunk_len_50__attr_"intercept"
35	agg_linear_trend__f_agg_"min"__chunk_len_50__attr_"rvalue"
36	agg_linear_trend__f_agg_"min"__chunk_len_50__attr_"slope"
37	agg_linear_trend__f_agg_"min"__chunk_len_50__attr_"stderr"
38	agg_linear_trend__f_agg_"min"__chunk_len_5__attr_"intercept"
39	agg_linear_trend__f_agg_"min"__chunk_len_5__attr_"rvalue"
40	agg_linear_trend__f_agg_"min"__chunk_len_5__attr_"slope"
41	agg_linear_trend__f_agg_"min"__chunk_len_5__attr_"stderr"
42	agg_linear_trend__f_agg_"var"__chunk_len_10__attr_"intercept"
43	agg_linear_trend__f_agg_"var"__chunk_len_10__attr_"rvalue"
44	agg_linear_trend__f_agg_"var"__chunk_len_10__attr_"slope"
45	agg_linear_trend__f_agg_"var"__chunk_len_10__attr_"stderr"

TABLE C.4: The features extracted using TSFRESH.

Index	Name
46	agg_linear_trend__f_agg_"var"__chunk_len_50__attr_"intercept"
47	agg_linear_trend__f_agg_"var"__chunk_len_50__attr_"rvalue"
48	agg_linear_trend__f_agg_"var"__chunk_len_50__attr_"slope"
49	agg_linear_trend__f_agg_"var"__chunk_len_50__attr_"stderr"
50	agg_linear_trend__f_agg_"var"__chunk_len_5__attr_"intercept"
51	agg_linear_trend__f_agg_"var"__chunk_len_5__attr_"rvalue"
52	agg_linear_trend__f_agg_"var"__chunk_len_5__attr_"slope"
53	agg_linear_trend__f_agg_"var"__chunk_len_5__attr_"stderr"
54	ar_coefficient__k_10__coeff_0
⋮	⋮
57	ar_coefficient__k_10__coeff_3
58	ar_coefficient__k_10__coeff_4
59	augmented_dickey_fuller__attr_"pvalue"
60	augmented_dickey_fuller__attr_"teststat"
61	augmented_dickey_fuller__attr_"usedlag"
62	autocorrelation__lag_0
⋮	⋮
71	autocorrelation__lag_9
72	binned_entropy__max_bins_10
73	c3__lag_1
74	c3__lag_2
75	c3__lag_3
76	change_quantiles__f_agg_"mean"__isabs_False__qh_0.2__ql_0.0
77	change_quantiles__f_agg_"mean"__isabs_False__qh_0.2__ql_0.2
78	change_quantiles__f_agg_"mean"__isabs_False__qh_0.2__ql_0.4
79	change_quantiles__f_agg_"mean"__isabs_False__qh_0.2__ql_0.6
80	change_quantiles__f_agg_"mean"__isabs_False__qh_0.2__ql_0.8
81	change_quantiles__f_agg_"mean"__isabs_False__qh_0.4__ql_0.0
82	change_quantiles__f_agg_"mean"__isabs_False__qh_0.4__ql_0.2
83	change_quantiles__f_agg_"mean"__isabs_False__qh_0.4__ql_0.4
84	change_quantiles__f_agg_"mean"__isabs_False__qh_0.4__ql_0.6
85	change_quantiles__f_agg_"mean"__isabs_False__qh_0.4__ql_0.8

TABLE C.4: The features extracted using TSFRESH.

Index	Name
86	change_quantiles__f_agg_"mean"__isabs_False__qh_0.6__ql_0.0
87	change_quantiles__f_agg_"mean"__isabs_False__qh_0.6__ql_0.2
88	change_quantiles__f_agg_"mean"__isabs_False__qh_0.6__ql_0.4
89	change_quantiles__f_agg_"mean"__isabs_False__qh_0.6__ql_0.6
90	change_quantiles__f_agg_"mean"__isabs_False__qh_0.6__ql_0.8
91	change_quantiles__f_agg_"mean"__isabs_False__qh_0.8__ql_0.0
92	change_quantiles__f_agg_"mean"__isabs_False__qh_0.8__ql_0.2
93	change_quantiles__f_agg_"mean"__isabs_False__qh_0.8__ql_0.4
94	change_quantiles__f_agg_"mean"__isabs_False__qh_0.8__ql_0.6
95	change_quantiles__f_agg_"mean"__isabs_False__qh_0.8__ql_0.8
96	change_quantiles__f_agg_"mean"__isabs_False__qh_1.0__ql_0.0
97	change_quantiles__f_agg_"mean"__isabs_False__qh_1.0__ql_0.2
98	change_quantiles__f_agg_"mean"__isabs_False__qh_1.0__ql_0.4
99	change_quantiles__f_agg_"mean"__isabs_False__qh_1.0__ql_0.6
100	change_quantiles__f_agg_"mean"__isabs_False__qh_1.0__ql_0.8
101	change_quantiles__f_agg_"mean"__isabs_True__qh_0.2__ql_0.0
102	change_quantiles__f_agg_"mean"__isabs_True__qh_0.2__ql_0.2
103	change_quantiles__f_agg_"mean"__isabs_True__qh_0.2__ql_0.4
104	change_quantiles__f_agg_"mean"__isabs_True__qh_0.2__ql_0.6
105	change_quantiles__f_agg_"mean"__isabs_True__qh_0.2__ql_0.8
106	change_quantiles__f_agg_"mean"__isabs_True__qh_0.4__ql_0.0
107	change_quantiles__f_agg_"mean"__isabs_True__qh_0.4__ql_0.2
108	change_quantiles__f_agg_"mean"__isabs_True__qh_0.4__ql_0.4
109	change_quantiles__f_agg_"mean"__isabs_True__qh_0.4__ql_0.6
110	change_quantiles__f_agg_"mean"__isabs_True__qh_0.4__ql_0.8
111	change_quantiles__f_agg_"mean"__isabs_True__qh_0.6__ql_0.0
112	change_quantiles__f_agg_"mean"__isabs_True__qh_0.6__ql_0.2
113	change_quantiles__f_agg_"mean"__isabs_True__qh_0.6__ql_0.4
114	change_quantiles__f_agg_"mean"__isabs_True__qh_0.6__ql_0.6
115	change_quantiles__f_agg_"mean"__isabs_True__qh_0.6__ql_0.8
116	change_quantiles__f_agg_"mean"__isabs_True__qh_0.8__ql_0.0
117	change_quantiles__f_agg_"mean"__isabs_True__qh_0.8__ql_0.2
118	change_quantiles__f_agg_"mean"__isabs_True__qh_0.8__ql_0.4

TABLE C.4: The features extracted using TSFRESH.

Index	Name
119	change_quantiles__f_agg_"mean"__isabs_True__qh_0.8__ql_0.6
120	change_quantiles__f_agg_"mean"__isabs_True__qh_0.8__ql_0.8
121	change_quantiles__f_agg_"mean"__isabs_True__qh_1.0__ql_0.0
122	change_quantiles__f_agg_"mean"__isabs_True__qh_1.0__ql_0.2
123	change_quantiles__f_agg_"mean"__isabs_True__qh_1.0__ql_0.4
124	change_quantiles__f_agg_"mean"__isabs_True__qh_1.0__ql_0.6
125	change_quantiles__f_agg_"mean"__isabs_True__qh_1.0__ql_0.8
126	change_quantiles__f_agg_"var"__isabs_False__qh_0.2__ql_0.0
127	change_quantiles__f_agg_"var"__isabs_False__qh_0.2__ql_0.2
128	change_quantiles__f_agg_"var"__isabs_False__qh_0.2__ql_0.4
129	change_quantiles__f_agg_"var"__isabs_False__qh_0.2__ql_0.6
130	change_quantiles__f_agg_"var"__isabs_False__qh_0.2__ql_0.8
131	change_quantiles__f_agg_"var"__isabs_False__qh_0.4__ql_0.0
132	change_quantiles__f_agg_"var"__isabs_False__qh_0.4__ql_0.2
133	change_quantiles__f_agg_"var"__isabs_False__qh_0.4__ql_0.4
134	change_quantiles__f_agg_"var"__isabs_False__qh_0.4__ql_0.6
135	change_quantiles__f_agg_"var"__isabs_False__qh_0.4__ql_0.8
136	change_quantiles__f_agg_"var"__isabs_False__qh_0.6__ql_0.0
137	change_quantiles__f_agg_"var"__isabs_False__qh_0.6__ql_0.2
138	change_quantiles__f_agg_"var"__isabs_False__qh_0.6__ql_0.4
139	change_quantiles__f_agg_"var"__isabs_False__qh_0.6__ql_0.6
140	change_quantiles__f_agg_"var"__isabs_False__qh_0.6__ql_0.8
141	change_quantiles__f_agg_"var"__isabs_False__qh_0.8__ql_0.0
142	change_quantiles__f_agg_"var"__isabs_False__qh_0.8__ql_0.2
143	change_quantiles__f_agg_"var"__isabs_False__qh_0.8__ql_0.4
144	change_quantiles__f_agg_"var"__isabs_False__qh_0.8__ql_0.6
145	change_quantiles__f_agg_"var"__isabs_False__qh_0.8__ql_0.8
146	change_quantiles__f_agg_"var"__isabs_False__qh_1.0__ql_0.0
147	change_quantiles__f_agg_"var"__isabs_False__qh_1.0__ql_0.2
148	change_quantiles__f_agg_"var"__isabs_False__qh_1.0__ql_0.4
149	change_quantiles__f_agg_"var"__isabs_False__qh_1.0__ql_0.6
150	change_quantiles__f_agg_"var"__isabs_False__qh_1.0__ql_0.8
151	change_quantiles__f_agg_"var"__isabs_True__qh_0.2__ql_0.0

TABLE C.4: The features extracted using TSFRESH.

Index	Name
152	change_quantiles__f_agg_"var"__isabs_True__qh_0.2__ql_0.2
153	change_quantiles__f_agg_"var"__isabs_True__qh_0.2__ql_0.4
154	change_quantiles__f_agg_"var"__isabs_True__qh_0.2__ql_0.6
155	change_quantiles__f_agg_"var"__isabs_True__qh_0.2__ql_0.8
156	change_quantiles__f_agg_"var"__isabs_True__qh_0.4__ql_0.0
157	change_quantiles__f_agg_"var"__isabs_True__qh_0.4__ql_0.2
158	change_quantiles__f_agg_"var"__isabs_True__qh_0.4__ql_0.4
159	change_quantiles__f_agg_"var"__isabs_True__qh_0.4__ql_0.6
160	change_quantiles__f_agg_"var"__isabs_True__qh_0.4__ql_0.8
161	change_quantiles__f_agg_"var"__isabs_True__qh_0.6__ql_0.0
162	change_quantiles__f_agg_"var"__isabs_True__qh_0.6__ql_0.2
163	change_quantiles__f_agg_"var"__isabs_True__qh_0.6__ql_0.4
164	change_quantiles__f_agg_"var"__isabs_True__qh_0.6__ql_0.6
165	change_quantiles__f_agg_"var"__isabs_True__qh_0.6__ql_0.8
166	change_quantiles__f_agg_"var"__isabs_True__qh_0.8__ql_0.0
167	change_quantiles__f_agg_"var"__isabs_True__qh_0.8__ql_0.2
168	change_quantiles__f_agg_"var"__isabs_True__qh_0.8__ql_0.4
169	change_quantiles__f_agg_"var"__isabs_True__qh_0.8__ql_0.6
170	change_quantiles__f_agg_"var"__isabs_True__qh_0.8__ql_0.8
171	change_quantiles__f_agg_"var"__isabs_True__qh_1.0__ql_0.0
172	change_quantiles__f_agg_"var"__isabs_True__qh_1.0__ql_0.2
173	change_quantiles__f_agg_"var"__isabs_True__qh_1.0__ql_0.4
174	change_quantiles__f_agg_"var"__isabs_True__qh_1.0__ql_0.6
175	change_quantiles__f_agg_"var"__isabs_True__qh_1.0__ql_0.8
176	cid_ce__normalize_False
177	cid_ce__normalize_True
178	count_above_mean
179	count_below_mean
180	cwt_coefficients__widths_(2, 5, 10, 20)__coeff_0__w_10
181	cwt_coefficients__widths_(2, 5, 10, 20)__coeff_0__w_2
182	cwt_coefficients__widths_(2, 5, 10, 20)__coeff_0__w_20
183	cwt_coefficients__widths_(2, 5, 10, 20)__coeff_0__w_5

TABLE C.4: The features extracted using TSFRESH.

Index	Name
⋮	⋮
239	cwt_coefficients__widths_(2, 5, 10, 20)__coeff_14__w_5
240	energy_ratio_by_chunks__num_segments_10__segment_focus_0
241	energy_ratio_by_chunks__num_segments_10__segment_focus_1
⋮	⋮
249	energy_ratio_by_chunks__num_segments_10__segment_focus_9
250	fft_aggregated__aggtype_"centroid"
251	fft_aggregated__aggtype_"kurtosis"
252	fft_aggregated__aggtype_"skew"
253	fft_aggregated__aggtype_"variance"
254	fft_coefficient__coeff_0__attr_"abs"
255	fft_coefficient__coeff_0__attr_"angle"
256	fft_coefficient__coeff_0__attr_"imag"
257	fft_coefficient__coeff_0__attr_"real"
⋮	⋮
650	fft_coefficient__coeff_99__attr_"abs"
651	fft_coefficient__coeff_99__attr_"angle"
652	fft_coefficient__coeff_99__attr_"imag"
653	fft_coefficient__coeff_99__attr_"real"
654	first_location_of_maximum
655	first_location_of_minimum
656	friedrich_coefficients__m_3__r_30__coeff_0
657	friedrich_coefficients__m_3__r_30__coeff_1
658	friedrich_coefficients__m_3__r_30__coeff_2
659	friedrich_coefficients__m_3__r_30__coeff_3
660	has_duplicate
661	has_duplicate_max
662	has_duplicate_min
663	index_mass_quantile__q_0.1
⋮	⋮
670	index_mass_quantile__q_0.9

TABLE C.4: The features extracted using TSFRESH.

Index	Name
671	kurtosis
672	large_standard_deviation__r_0.05
673	large_standard_deviation__r_0.1
⋮	⋮
690	large_standard_deviation__r_0.95
691	last_location_of_maximum
692	last_location_of_minimum
693	length
694	linear_trend__attr_"intercept"
695	linear_trend__attr_"pvalue"
696	linear_trend__attr_"rvalue"
697	linear_trend__attr_"slope"
698	linear_trend__attr_"stderr"
699	longest_strike_above_mean
700	longest_strike_below_mean
701	max_langevin_fixed_point__m_3__r_30
702	maximum
703	mean
704	mean_abs_change
705	mean_change
706	mean_second_derivative_central
707	median
708	minimum
709	number_crossing_m__m_-1
710	number_crossing_m__m_0
711	number_crossing_m__m_1
712	number_cwt_peaks__n_1
713	number_cwt_peaks__n_5
714	number_peaks__n_1
715	number_peaks__n_10
716	number_peaks__n_3
717	number_peaks__n_5

TABLE C.4: The features extracted using TSFRESH.

Index	Name
718	number_peaks__n_50
719	partial_autocorrelation__lag_0
720	partial_autocorrelation__lag_1
⋮	⋮
728	partial_autocorrelation__lag_9
729	percentage_of_reoccurring_datapoints_to_all_datapoints
730	percentage_of_reoccurring_values_to_all_values
731	quantile__q_0.1
732	quantile__q_0.2
⋮	⋮
738	quantile__q_0.9
739	range_count__max_1__min_-1
740	ratio_beyond_r_sigma__r_0.5
741	ratio_beyond_r_sigma__r_1
742	ratio_beyond_r_sigma__r_1.5
743	ratio_beyond_r_sigma__r_10
744	ratio_beyond_r_sigma__r_2
745	ratio_beyond_r_sigma__r_2.5
746	ratio_beyond_r_sigma__r_3
747	ratio_beyond_r_sigma__r_5
748	ratio_beyond_r_sigma__r_6
749	ratio_beyond_r_sigma__r_7
750	ratio_value_number_to_time_series_length
751	skewness
752	spkt_welch_density__coeff_2
753	spkt_welch_density__coeff_5
754	spkt_welch_density__coeff_8
755	standard_deviation
756	sum_of_reoccurring_data_points
757	sum_of_reoccurring_values
758	sum_values
759	symmetry_looking__r_0.0

TABLE C.4: The features extracted using TSFRESH.

Index	Name
760	symmetry_looking__r_0.05
⋮	⋮
778	symmetry_looking__r_0.95
779	time_reversal_asymmetry_statistic__lag_1
780	time_reversal_asymmetry_statistic__lag_2
781	time_reversal_asymmetry_statistic__lag_3
782	value_count__value_-inf
783	value_count__value_0
784	value_count__value_1
785	value_count__value_inf
786	value_count__value_nan
787	variance
788	variance_larger_than_standard_deviation

BIBLIOGRAPHY

Aas, K., C. Czado, A. Frigessi, and H. Bakkend (2009). Pair-copula constructions of multiple dependence. *Insurance: Mathematics and Economics 44*(2), 182–198.

Abdallah, I., C. Lataniotis, and B. Sudret (2019). Parametric hierarchical kriging for multi-fidelity aero-servo-elastic simulators—application to extreme loads on wind turbines. *Prob. Eng. Mech. 55*, 67–77.

Abdelaziz, A. H., S. Watanabe, J. R. Hershey, E. Vincent, and D. Kolossa (2015). Uncertainty propagation through deep neural networks. In *Interspeech 2015*.

Agbayani, N. A. (2010). Defects, damage, and repairs subject to high-cycle fatigue: Examples from wind farm tower design. In *Forensic Engineering 2009: Pathology of the Built Environment*, pp. 546–555.

Alam, M. A. and K. Fukumizu (2014). Hyperparameter selection in kernel principal component analysis. *Journal of Computer Science 10(7)*, 1139–1150.

Ang, A. and W. Tang (2007). *Probability concepts in engineering: emphasis on applications to civil and environmental engineering*. Wiley.

Arlot, S. and A. Celisse (2010). A survey of cross-validation procedures for model selection. *Statistics Surveys 4*, 40–79.

ASTM E1049-85 (2017). Standard Practices for Cycle Counting in Fatigue Analysis. Technical report, ASTM International.

Bachoc, F. (2013). Cross validation and maximum likelihood estimations of hyper-parameters of Gaussian processes with model misspecifications. *Comput. Stat. Data Anal. 66*, 55–69.

Bachoc, F., G. Bois, J. Garnier, and J.-M. Martinez (2014). Calibration and improved prediction of computer models by universal Kriging. *Nucl. Sci. Eng. 176*, 91–97.

Bect, J., E. Vazquez, et al. (2014). STK: a Small (Matlab/Octave) Toolbox for Kriging. Release 2.4.

Bedford, T., R. M. Cooke, et al. (2002). Vines–a new graphical model for dependent random variables. *The Annals of Statistics 30*(4), 1031–1068.

Belkin, M. and P. Niyogi (2002). Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pp. 585–591.

Bengio, Y., J.-f. Paiement, P. Vincent, O. Delalleau, N. L. Roux, and M. Ouimet (2004). Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. In *Advances in neural information processing systems*, pp. 177–184.

Berglind, B., J. d. Jesus, and R. Wisniewski (2016). Representation of fatigue for wind turbine control. *Wind Energy 19*(12), 2189–2203.

Bertsekas, D. P. (1999). *Nonlinear programming*. Athena scientific Belmont.

Berveiller, M., B. Sudret, and M. Lemaire (2006). Stochastic finite elements: a non intrusive approach by regression. *Eur. J. Comput. Mech. 15*(1-3), 81–92.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag.

Blatman, G. (2009). *Adaptive sparse polynomial chaos expansions for uncertainty propagation and sensitivity analysis*. Ph. D. thesis, Université Blaise Pascal, Clermont-Ferrand.

Blatman, G. and B. Sudret (2010). An adaptive algorithm to build up sparse polynomial chaos expansions for stochastic finite element analysis. *Prob. Eng. Mech. 25*, 183–197.

Blatman, G. and B. Sudret (2011). Adaptive sparse polynomial chaos expansion based on Least Angle Regression. *J. Comput. Phys 230*, 2345–2367.

Boureau, Y.-l., Y. L. Cun, et al. (2008). Sparse feature learning for deep belief networks. In *Advances in neural information processing systems*, pp. 1185–1192.

Calandra, R., J. Peters, C. E. Rasmussen, and M. P. Deisenroth (2016). Manifold Gaussian processes for regression. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pp. 3338–3345. IEEE.

Camastra, F. (2003). Data dimensionality estimation methods: a survey. *Pattern recognition 36*(12), 2945–2954.

Chen, M., K. Weinberger, F. Sha, and Y. Bengio (2014). Marginalized denoising auto-encoders for nonlinear representations. In *International Conference on Machine Learning*, pp. 1476–1484.

Chevreuil, M., R. Lebrun, A. Nouy, and P. Rai (2015). A least-squares method for sparse low rank approximation of multivariate functions. *SIAM/ASA J. Uncertainty Quantification 3*(1), 897–921.

Chkifa, A., A. Cohen, G. Migliorati, F. Nobile, and R. Tempone (2015). Discrete least squares polynomial approximation with random evaluations – application to parametric and stochastic elliptic PDEs. *ESAIM: Mathematical Modelling and Numerical Analysis 49*(3), 815–837.

Christ, M., N. Braun, J. Neuffer, and A. W. Kempa-Liehr (2018). Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). *Neurocomputing 307*, 72–77. Documentation: https://tsfresh.readthedocs.io.

Constantine, P. G., E. Dow, and Q. Wang (2014). Active subspace methods in theory and practice: applications to kriging surfaces. *SIAM Journal on Scientific Computing 36*(4), A1500–A1524.

Couckuyt, I., T. Dhaene, and P. Demeester (2014). ooDACE Toolbox: A Flexible Object-Oriented Kriging Implementation. *Journal of Machine Learning Research 15*, 3183–3186.

Cox, T. F. and M. Cox (2000). *Multidimensional Scaling, Second Edition* (2 ed.). Chapman and Hall/CRC.

Cressie, N. A. C. (1993). Statistics for Spatial Data. In *Statistics for Spatial Data*, pp. 1–26. John Wiley & Sons, Inc.

Damianou, A. and N. Lawrence (2013). Deep Gaussian processes. In *Artificial Intelligence and Statistics*, pp. 207–215.

Dashti, M. and A. M. Stuart (2017). The Bayesian approach to inverse problems. In *Handbook of Uncertainty Quantification*, pp. 311–428. Springer International Publishing.

de Rocquigny, E. (2006a). La maîtrise des incertitudes dans un contexte industriel : 1^e partie – Une approche méthodologique globale basée sur des exemples. *J. Soc. Française Stat. 147*(3), 33–71.

de Rocquigny, E. (2006b). La maîtrise des incertitudes dans un contexte industriel : 2^e partie – Revue des méthodes de modélisation statistique, physique et numérique. *J. Soc. Française Stat. 147*(3), 73–106.

De Rocquigny, E., N. Devictor, and S. Tarantola (Eds.) (2008). *Uncertainty in industrial practice – A guide to quantitative uncertainty management*. John Wiley & Sons.

Deman, G., K. Konakli, B. Sudret, J. Kerrou, P. Perrochet, and H. Benabderrahmane (2016). Using sparse polynomial chaos expansions for the global sensitivity analysis of groundwater lifetime expectancy in a multi-layered hydrogeological model. *Reliab. Eng. Sys. Safety 147*, 156–169.

Deutsch, C. V., A. G. Journel, et al. (1992). Geostatistical software library and user's guide. *New York 119*, 147.

Dheeru, D. and E. Karra Taniskidou (2017). UCI machine learning repository. http://archive.ics.uci.edu/ml.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik 1*, 269–271.

Ditlevsen, O. and H. Madsen (1996). *Structural reliability methods*. J. Wiley and Sons, Chichester.

Djolonga, J., A. Krause, and V. Cevher (2013). High-dimensional Gaussian process bandits. In *Advances in Neural Information Processing Systems*, pp. 1025–1033.

Dubrule, O. (1983). Cross validation of Kriging in a unique neighborhood. *J. Int. Assoc Math. Geology 15*(6), 687–699.

Dupuy, D., C. Helbert, and J. Franco (2015). DiceDesign and DiceEval: Two R packages for design and analysis of computer experiments. *Journal of Statistical Software 65*(11), 1–38.

Durrande, N., D. Ginsbourger, and O. Roustant (2012). Additive covariance kernels for high-dimensional Gaussian process modeling. In *Annales de la Faculté de Sciences de Toulouse*, Volume 21, pp. p–481.

Echard, B., N. Gayton, and M. Lemaire (2011). AK-MCS: an active learning reliability method combining Kriging and Monte Carlo simulation. *Structural Safety 33*(2), 145–154.

Efron, B., T. Hastie, I. Johnstone, and R. Tibshirani (2004). Least angle regression. *Annals of Statistics 32*, 407–499.

Eldred, M. S., L. W. T. Ng, M. F. Barone, and S. P. Domino (2016). Multifidelity uncertainty quantification using spectral stochastic discrepancy models. In *Handbook of Uncertainty Quantification*. Springer International Publishing.

European, commission (2011). Energy road map 2050.

Fang, K.-T., R. Li, and A. Sudjianto (2005). *Design and modeling for computer experiments*. CRC Press.

Floyd, R. W. (1962). Algorithm 97: Shortest path. *Commun. ACM 5*(6), 345.

Fornasier, M., K. Schnass, and J. Vybiral (2012). Learning functions of few arbitrary linear parameters in high dimensions. *Foundations of Computational Mathematics 12*(2), 229–262.

Forrester, A., A. Sobester, and A. Keane (2008). *Engineering design via surrogate modelling: a practical guide*. Wiley.

Fukunaga, K. (2013). *Introduction to statistical pattern recognition*. Academic press.

Gautschi, W. (2004). *Orthogonal Polynomials: Computation and Approximation*. Numerical Mathematics and Scientific Computation. Oxford University Press.

Ghanem, R. and P. Spanos (1991). *Stochastic finite elements – A spectral approach*. Springer Verlag, New York. (Reedited by Dover Publications, Mineola, 2003).

Ghiocel, D. and R. Ghanem (2002). Stochastic finite element analysis of seismic soil-structure interaction. *J. Eng. Mech. 128*, 66–77.

Ghodsi, A. (2006). Dimensionality reduction a short tutorial. Technical report, Department of Statistics and Actuarial Science, Univ. of Waterloo, Ontario, Canada.

Glorot, X., A. Bordes, and Y. Bengio (2011). Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 513–520.

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning*. MIT Press.

Goodfellow, I., H. Lee, Q. V. Le, A. Saxe, and A. Y. Ng (2009). Measuring invariances in deep networks. In *Advances in neural information processing systems*, pp. 646–654.

GPy (2012). GPy: A Gaussian process framework in python. http://github.com/SheffieldML/GPy.

Grasse, F., V. Trappe, S. Thöns, and S. Said (2011). Structural health monitoring of wind turbine blades by strain measurement and vibration analysis.

Günter, S., N. N. Schraudolph, and S. V. N. Vishwanathan (2007). Fast iterative kernel principal component analysis. *J. Mach. Learn. Res. 8*, 1893–1918.

Hagan, M. T., H. B. Demuth, M. H. Beale, and O. De Jesús (1996). *Neural network design*, Volume 20. Pws Pub. Boston.

Halko, N., P.-G. Martinsson, Y. Shkolnisky, and M. Tygert (2011). An algorithm for the principal component analysis of large data sets. *SIAM Journal on Scientific computing 33*(5), 2580–2594.

Han, Z., R. Zimmerman, and S. Görtz (2012). Alternative cokriging method for variable-fidelity surrogate modeling. *AIAA journal 50*(5), 1205–1210.

Hastie, T., J. Taylor, R. Tibshirani, and G. Walther (2007). Forward stagewise regression and the monotone lasso. *Electronic Journal of Statistics 1*, 1–29.

Hastie, T., R. Tibshirani, and J. Friedman (2001). *The elements of statistical learning: Data mining, inference and prediction*. Springer, New York.

Helton, J. C., R. L. Iman, and J. B. Brown (1985). Sensitivity analysis of the asymptotic behavior of a model for the environmental movement of radionuclides. *Ecological modelling 28*(4), 243–278.

Hinton, G. E. (2012). A practical guide to training restricted boltzmann machines. In *Neural Networks: Tricks of the Trade (2nd ed.)*, Volume 7700 of *Lecture Notes in Computer Science*, pp. 599–619. Springer.

Hinton, G. E., S. Osindero, and Y.-W. Teh (2006). A fast learning algorithm for deep belief nets. *Neural computation 18*(7), 1527–1554.

Hinton, G. E. and R. R. Salakhutdinov (2006). Reducing the dimensionality of data with neural networks. *Science 313*(5786), 504–507.

Hoffmann, H. (2007). Kernel PCA for novelty detection. *Pattern Recogn. 40*(3), 863–874.

Hornik, K., M. Stinchcombe, and H. White (1989). Multilayer feedforward networks are universal approximators. *Neural networks 2*(5), 359–366.

Huang, W.-b., D. Zhao, F. Sun, H. Liu, and E. Y. Chang (2015). Scalable Gaussian process regression using deep neural networks. In *IJCAI*, pp. 3576–3582.

Hyvärinen, A. and E. Oja (1997). One-unit learning rules for independent component analysis. In *Advances in neural information processing systems*, pp. 480–486.

Iooss, B. and P. Lemaître (2015a). A review on global sensitivity analysis methods. In *Uncertainty management in simulation-optimization of complex systems*, pp. 101–122. Springer.

Iooss, B. and P. Lemaître (2015b). A review on global sensitivity analysis methods. In *Uncertainty Management in Simulation-Optimization of Complex Systems*, pp. 101–122. Springer.

Jakeman, J., M. Eldred, and K. Sargsyan (2015). Enhancing ℓ_1-minimization estimates of polynomial chaos expansions using basis selection. *J. Comput. Phys. 289*, 18–34.

Joe, H. (Ed.) (2015). *Dependence modeling with copulas*. CRC Press.

Jonkman, B. J. (2009). Turbsim user's guide: Version 1.50. Technical report, National Renewable Energy Lab.(NREL), Golden, CO (United States).

Jonkman, J. (2013). The new modularization framework for the fast wind turbine cae tool. In *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, pp. 202.

Jonkman, J., S. Butterfield, W. Musial, and G. Scott (2009). Definition of a 5-mw reference wind turbine for offshore system development. Technical report, National Renewable Energy Lab.(NREL), Golden, CO (United States).

Kauzlarich, J. (1989). The palmgren-miner rule derived. In *Tribology Series*, Volume 14, pp. 175–179. Elsevier.

Keese, A. and H.-G. Matthies (2005). Hierarchical parallelisation for the solution of stochastic finite element equations. *Computers & Structures 83*, 1033–1047.

Kersaudy, P., B. Sudret, N. Varsier, O. Picon, and J. Wiart (2015). A new surrogate modeling technique combining Kriging and polynomial chaos expansions – Application to uncertainty analysis in computational dosimetry. *J. Comput. Phys 286*, 103–117.

Knockaert, L., B. De Backer, and D. De Zutter (1999). SVD compression, unitary transforms, and computational complexity. *IEEE transactions on signal processing 47*(10), 2724–2729.

Konakli, K. and B. Sudret (2016a). Global sensitivity analysis using low-rank tensor approximations. *Reliab. Eng. Sys. Safety 156*, 64–83.

Konakli, K. and B. Sudret (2016b). Polynomial meta-models with canonical low-rank approximations: Numerical insights and comparison to sparse polynomial chaos expansions. *J. Comput. Phys. 321*, 1144–1169.

Konakli, K., B. Sudret, J. Kerrou, P. Perrochet, and H. Benabderrahmane (2016). Reliability analysis of high-dimensional models using low-rank tensor approximations. *Prob. Eng. Mech. 46*, 18–36.

Koomey, J., S. Berard, M. Sanchez, and H. Wong (2011). Implications of historical trends in the electrical efficiency of computing. *IEEE Annals of the History of Computing 33*(3), 46–54.

Krige, D. G. (1951). A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of the Southern African Institute of Mining and Metallurgy 52*(6), 119–139.

Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. Curran Associates, Inc.

Kuzmin, D. and M. K. Warmuth (2007). Online kernel pca with entropic matrix updates. In *ICML*, Volume 227 of *ACM International Conference Proceeding Series*, pp. 465–472. ACM.

Kwok, J. T. and I. W. Tsang (2003). The pre-image problem in kernel methods. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 408–415.

Lataniotis, C., S. Marelli, and B. Sudret (2018). The Gaussian process modelling module in UQLab. *Soft Comput. Civil Eng. 2*(3), 91–116.

Lataniotis, C., S. Marelli, and B. Sudret (2019). UQLab user manual – Kriging (Gaussian process modelling). Technical report, Chair of Risk, Safety & Uncertainty Quantification, ETH Zurich. Report # UQLab-V1.2-105.

Lataniotis, C., E. Torre, S. Marelli, and B. Sudret (2019). UQLab user manual – The Input module. Technical report, Chair of Risk, Safety & Uncertainty Quantification, ETH Zurich. Report # UQLab-V1.2-102.

Lawrence, N. (2005). Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *Journal of Machine Learning Research 6*, 1783–1816.

Le Gratiet, L., S. Marelli, and B. Sudret (2016). *Metamodel-based sensitivity analysis: polynomial chaos expansions and Gaussian processes*, Chapter 8. Springer.

Le Maître, O., M. Reagan, H. Najm, R. Ghanem, and O. Knio (2002). A stochastic projection method for fluid flow – II. Random process. *J. Comput. Phys. 181*, 9–44.

Lebrun, R. and A. Dutfoy (2009). A generalization of the Nataf transformation to distributions with elliptical copula. *Probabilistic Engineering Mechanics 24*(2), 172–178.

Lemaire, M. (2009). *Structural reliability*. Wiley.

Leshno, M., V. Y. Lin, A. Pinkus, and S. Schocken (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks 6*(6), 861–867.

Li, C. and A. Der Kiureghian (1993). Optimal discretization of random fields. *J. Eng. Mech. 119*(6), 1136–1154.

Lima, A., H. Zen, Y. Nankaku, C. Miyajima, K. Tokuda, and T. Kitamura (2003). On the use of kernel PCA for feature extraction in speech recognition. In *INTERSPEECH*. ISCA.

Lin, J. (1991). Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory 37*(1), 145–151.

Lophaven, S. N., H. B. Nielsen, and J. Sondergaard (2002). Aspects of the Matlab toolbox DACE. Technical report, Informatics and Mathematical Modelling.

Marelli, S., C. Lamas, K. Konakli, C. Mylonas, P. Wiederkehr, and B. Sudret (2019). UQLab user manual – Sensitivity analysis. Technical report, Chair of Risk, Safety & Uncertainty Quantification, ETH Zurich. Report # UQLab-V1.2-106.

Marelli, S., R. Schöbi, and B. Sudret (2019). UQLab user manual – Structural Reliability. Technical report, Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich. Report # UQLab-V1.2-107.

Marelli, S. and B. Sudret (2014). UQLab: A framework for uncertainty quantification in Matlab. In *Vulnerability, Uncertainty, and Risk (Proc. 2nd Int. Conf. on Vulnerability, Risk Analysis and Management (ICVRAM2014), Liverpool, United Kingdom)*, pp. 2554–2563.

Marelli, S. and B. Sudret (2018). UQLab user manual – polynomial chaos expansions. Technical report, Chair of Risk, Safety & Uncertainty Quantification, ETH Zurich. Report UQLab-V1.1-104.

Marrel, A., B. Iooss, F. Van Dorpe, and E. Volkova (2008). An efficient methodology for modeling complex computer codes with Gaussian processes. *Computational Statistics & Data Analysis* 52(10), 4731–4744.

MathWorks Inc. (2017). MATLAB and Statistics and Machine Learning Toolbox Release 2017b.

McCulloch, W. S. and W. Pitts (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics* 5(4), 115–133.

McKay, M. D., R. J. Beckman, and W. J. Conover (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* 2, 239–245.

Melchers, R.-E. (1999). *Structural reliability analysis and prediction*. John Wiley & Sons.

Miner, M. A. (1945). Cumulative damage in fatigue. *J. Appl. Mech.* (9), 159–164.

Ming-Hsuan Yang, Ahuja, N. and D. Kriegman (2000). Face recognition using kernel eigenfaces. In *Image Processing, 2000. Proceedings. 2000 International Conference on*, Volume 1, pp. 37–40 vol.1.

Montufar, G. F., R. Pascanu, K. Cho, and Y. Bengio (2014). On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pp. 2924–2932.

Moustapha, M., B. Sudret, J.-M. Bourinet, and B. Guillaume (2016). Quantile-based optimization under uncertainties using adaptive Kriging surrogate models. *Struct. Multidisc. Optim.*.

Moustapha, M., B. Sudret, J.-M. Bourinet, and B. Guillaume (2018). Comparative study of Kriging and support vector regression for structural engineering applications. *ASCE-ASME J. Risk Uncertainty Eng. Syst., Part A: Civ. Eng.* 4(2). Paper #04018005.

Murphy, K. P. (2012). Machine learning: A probabilistic perspective. adaptive computation and machine learning.

Nataf, A. (1962). Détermination des distributions dont les marges sont données. *C. R. Acad. Sci. Paris 225*, 42–43.

Nelsen, R. B. (2006). *An introduction to copulas* (2nd ed.), Volume 139 of *Lecture Notes in Statistics*. Springer-Verlag, New York.

Olshausen, B. A. and D. J. Field (1997). Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research 37*(23), 3311–3325.

Partridge, M. and R. A. Calvo (1998). Fast dimensionality reduction and simple PCA. *Intelligent Data Analysis 2*(1–4), 203 – 214.

Parzen, E. (1962). On estimation of a probability density function and mode. *The Annals of Mathematical Statistics 33*(3), 1065–1076.

Paulson, C. and G. Ragkousis (2015). pyKriging: A Python Kriging Toolkit. http://doi.org/10.5281/zenodo.21389.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *Phil. Mag. 6*(2), 559–572.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research 12*, 2825–2830.

Picheny, V., D. Ginsbourger, and O. Roustant (2016). *DiceOptim: Kriging-Based Optimization for Computer Experiments*. R package version 0.8-1.

Proakis, J. G. and D. G. Manolakis (1996). *Digital Signal Processing (3rd Ed.): Principles, Algorithms, and Applications*. Prentice-Hall, Inc.

Rasmussen, C. and C. Williams (2006). *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press.

Rasmussen, C. E. and H. Nickisch (2010). Gaussian Processes for Machine Learning (GPML) Toolbox. *Journal of Machine Learning Research 11*, 3011–3015.

Ricker, N. (1944). Wavelet functions and their polynomials. *Geophysics 9*(3), 314–323.

Rifai, S., P. Vincent, X. Muller, X. Glorot, and Y. Bengio (2011). Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pp. 833–840. Omnipress.

Ripley, B. D. (2009). *Stochastic simulation*, Volume 316. John Wiley & Sons.

Rokhlin, V., A. Szlam, and M. Tygert (2009). A randomized algorithm for principal component analysis. *SIAM Journal on Matrix Analysis and Applications 31*(3), 1100–1124.

Rosenblatt, M. (1952). Remarks on a multivariate transformation. *The Annals of Mathematical Statistics 23*, 470–472.

Rosenblatt, M. (1956). Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, 832–837.

Roustant, O., D. Ginsbourger, and Y. Deville (2012). DiceKriging, DiceOptim : Two R Packages for the Analysis of Computer Experiments by Kriging-Based Metamodeling and Optimization. *Journal of Statistical Software 51*(1).

Roweis, S. T. and L. K. Saul (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science 290*, 2323–2326.

Rychlik, I. (1987). A new definition of the rainflow cycle counting method. *International journal of fatigue 9*(2), 119–121.

Sacks, J., W. Welch, T. Mitchell, and H. Wynn (1989). Design and analysis of computer experiments. *Stat. Sci. 4*, 409–435.

Saltelli, A., K. Chan, and E. Scott (Eds.) (2000). *Sensitivity analysis*. J. Wiley & Sons.

Santner, T. J., B. J. Williams, and W. I. Notz (2003). *The Design and Analysis of Computer Experiments*. Springer New York.

Schöbi, R., S. Marelli, and B. Sudret (2019). UQLab user manual – Polynomial chaos Kriging. Technical report, Chair of Risk, Safety & Uncertainty Quantification, ETH Zurich. Report # UQLab-V1.2-109.

Schöbi, R., B. Sudret, and J. Wiart (2015). Polynomial-chaos-based Kriging. *Int. J. Uncertainty Quantification 5*(2), 171–193.

Schölkopf, B., A. Smola, and K.-R. Müller (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput. 10*(5), 1299–1319.

Sheikholeslami, R., S. Razavi, H. V. Gupta, W. Becker, and A. Haghnegahdar (2019). Global sensitivity analysis for high-dimensional problems: How to objectively group factors and measure robustness and convergence while reducing computational cost. *Environmental modelling & software 111*, 282–299.

Silverman, B. W. (2018). *Density estimation for statistics and data analysis*. Routledge.

Sklar, A. (1959). Fonctions de répartition à n dimensions et leurs marges. *Publications de l'Institut de Statistique de L'Université de Paris 8 8*(1), 11.

Snoek, J., H. Larochelle, and R. P. Adams (2012). Practical Bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pp. 2951–2959.

Sobol', I. (1993). Sensitivity estimates for nonlinear mathematical models. *Math. Modeling & Comp. Exp. 1*, 407–414.

Soize, C. and R. Ghanem (2004). Physical systems with random uncertainties: chaos representations with arbitrary probability measure. *SIAM J. Sci. Comput. 26*(2), 395–410.

Soize, C. and R. Ghanem (2017). Polynomial chaos representation of databases on manifolds. *J. Comp. Phys. 335*, 201–221.

Stein, M. L. (1999). *Interpolation of Spatial Data: Some Theory for Kriging*. Springer New York.

Strange, H. and R. Zwiggelaar (2011). A generalised solution to the out-of-sample extension problem in manifold learning. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, pp. 293–296.

Sudret, B. (2007). Uncertainty propagation and sensitivity analysis in mechanical models - Contributions to structural reliability and stochastic

spectral methods. Habilitation thesis, Université Blaise Pascal, Clermont-Ferrand, France.

Sun, J., C. Fyfe, and M. Crowe (2012). Extending Sammon mapping with Bregman divergences. *Information Sciences 187*, 72 – 92.

Suresh, S. (1998). *Fatigue of materials*. Cambridge university press.

Tenenbaum, J. B., V. De Silva, and J. C. Langford (2000). A global geometric framework for nonlinear dimensionality reduction. *Science 290*(5500), 2319–2323.

Theis, L., W. Shi, A. Cunningham, and F. Huszár (2017). Lossy image compression with compressive autoencoders. *arXiv preprint arXiv:1703.00395*.

Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *J. Royal Stat. Soc., Series B 58*, 267–288.

Tibshirani, R. J. and R. Tibshirani (2009). A bias correction for the minimum error rate in cross-validation. *The Annals of Applied Statistics*, 822–829.

Tipping, M. E. (2001). Sparse kernel principal component analysis. In *Advances in Neural Information Processing Systems 13*, pp. 633–639. MIT Press.

Tipping, M. E. and C. M. Bishop (1999a). Mixtures of probabilistic principal component analyzers. *Neural computation 11*(2), 443–482.

Tipping, M. E. and C. M. Bishop (1999b). Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 61*(3), 611–622.

Torre, E., S. Marelli, P. Embrechts, and B. Sudret (2018). A general framework for data-driven uncertainty quantification under complex input dependencies using vine copulas. *Prob. Eng. Mech. 55*, 1–16.

Torre, E., S. Marelli, P. Embrechts, and B. Sudret (2019). Data-driven polynomial chaos expansion for machine learning regression. *J. Comp. Phys. 388*, 601–623.

Torre, E., S. Marelli, and B. Sudret (2019). UQLab user manual – Statistical inference. Technical report, Chair of Risk, Safety & Uncertainty Quantification, ETH Zurich. Report # UQLab-V1.3-103.

Tripathy, R., I. Bilionis, and M. Gonzalez (2016). Gaussian processes with built-in dimensionality reduction: Applications to high-dimensional uncertainty propagation. *J. Comp. Phys. 321*, 191–223.

Tsompanakis, Y., N. Lagaros, and M. Papadrakis (Eds.) (2008). *Structural design optimization considering uncertainties*. Taylor & Francis.

Vachon, W. (2002). Long-term o&m costs of wind turbines based on failure rates and repair costs. In *Proceedings WINDPOWER, American Wind Energy Association annual conference, Portland, OR*, pp. 2–5.

Van Der Maaten, L., E. Postma, and J. Van den Herik (2009). Dimensionality reduction: a comparative review. *J Mach Learn Res 10*, 66–71.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.

Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley-Interscience.

Verleysen, M. and D. François (2005). The curse of dimensionality in data mining and time series prediction. In J. Cabestany, A. Prieto, and F. Sandoval (Eds.), *Computational Intelligence and Bioinspired Systems*, Volume 3512 of *Lecture Notes in Computer Science*, pp. 758–770. Springer Berlin Heidelberg.

Vincent, P., H. Larochelle, Y. Bengio, and P.-A. Manzagol (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103. ACM.

Wahlström, N., T. B. Schön, and M. P. Deisenroth (2015). Learning deep dynamical models from image pixels. *IFAC-PapersOnLine 48*(28), 1059–1064.

Weinberger, K. Q., F. Sha, and L. K. Saul (2004). Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 106. ACM.

Welch, W., R. Buck, J. Sacks, H. Wynn, T. Mitchell, and M. Morris (1992). Screening, predicting, and computer experiments. *Technometrics 34*, 15–25.

Weston, J., B. Schölkopf, and G. H. Bakir (2004). Learning to find pre-images. In *Advances in neural information processing systems*, pp. 449–456.

Wilkes, J., J. Moccia, P. Wilczek, R. Gruet, V. Radvilaitè, and M. Dragan (2011). EU energy policy to 2050–achieving 80-95% emissions reduction. *Report by the European Wind Energy Association (EWEA)*.

Williams, C. K. (2002). On a connection between kernel PCA and metric multidimensional scaling. *Machine Learning 46*(1), 11–19.

Wilson, A. G., Z. Hu, R. Salakhutdinov, and E. P. Xing (2016). Deep kernel learning. In *Artificial Intelligence and Statistics*, pp. 370–378.

Wu, C. J. and M. Hamada (2000). Experiments: Planning. *Analysis, and Parameter Design Optimization. Wiley*.

Xie, J., L. Xu, and E. Chen (2012). Image denoising and inpainting with deep neural networks. In *Advances in neural information processing systems*, pp. 341–349.

Xiu, D. (2010). *Numerical methods for stochastic computations – A spectral method approach*. Princeton University press.

Xiu, D. and J. Hesthaven (2005). High-order collocation methods for differential equations with random inputs. *SIAM J. Sci. Comput. 27*(3), 1118–1139.

Xiu, D. and G. E. Karniadakis (2002). The Wiener-Askey polynomial chaos for stochastic differential equations. *SIAM J. Sci. Comput. 24*(2), 619–644.

Yang, X., W. Li, and A. Tartakovsky (2018). Sliced-inverse-regression–aided rotated compressive sensing method for uncertainty quantification. *SIAM/ASA Journal on Uncertainty Quantification 6*(4), 1532–1554.

Zhang, Z., T.-W. Weng, and L. Daniel (2017). Big-data tensor recovery for high-dimensional uncertainty quantification of process variations. *IEEE Transactions on Components, Packaging and Manufacturing Technology 7*(5), 687–697.