Uncertainty quantification with dependent input data

Including applications to offshore wind farms

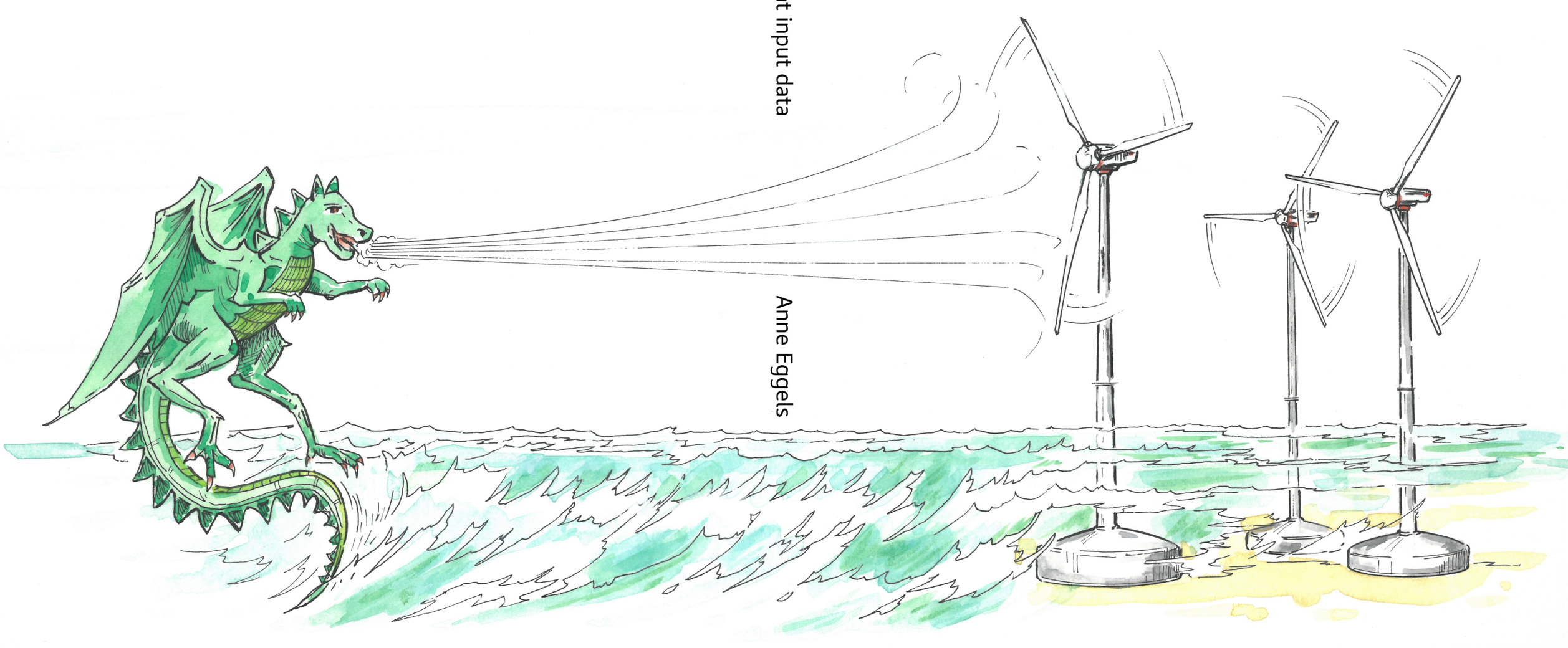
Eggels, A.W.

Uncertainty quantification with dependent input data

Anne Eggels

Uncertainty quantification with dependent input data

Anne Eggels



Uncertainty quantification with dependent input data

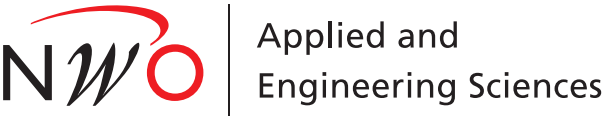
including applications to offshore wind farms

Anne Eggels

Cover design by J.H.M. Eggels

A catalogue record is available from UvA-DARE
ISBN 978-94-6323-848-9

Printed by Gildeprint, Enschede



This research is part of the EUROS programme, which is supported by NWO domain Applied and Engineering Sciences and partly funded by the Ministry of Economic Affairs. The research work was carried out at Centrum Wiskunde & Informatica (CWI), the Dutch national institute for mathematics and computer science.

Uncertainty quantification with dependent input data

including applications to offshore wind farms

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor

aan de Universiteit van Amsterdam

op gezag van de Rector Magnificus

prof. dr. ir. K.I.J. Maex

ten overstaan van een door het College voor Promoties ingestelde commissie,

in het openbaar te verdedigen in de Agnietenkapel

op woensdag 6 november 2019, te 10.00 uur

door Anne Willemien Eggels

geboren te Weert

Promotiecommissie:

Promotor:

prof. dr. D.T. Crommelin	Universiteit van Amsterdam

Copromotor:

prof. dr. ir. B. Koren	Technische Universiteit Eindhoven

Overige leden:

prof. dr. ir. H. Bijl	Universiteit Leiden
prof. dr. S.J. Watson	Technische Universiteit Delft
dr. ir. B. Sanderse	Centrum Wiskunde & Informatica
prof. dr. J.J.O.O. Wiegerinck	Universiteit van Amsterdam
dr. C.C. Stolk	Universiteit van Amsterdam
prof. dr. ir. A.G. Hoekstra	Universiteit van Amsterdam

Faculteit:

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

Summary

Uncertainty quantification with dependent input data - including applications to offshore wind farms

Offshore wind energy is an alternative power source which is used among others to increase the production of energy from renewable sources in the Netherlands. Due to an increasing number of turbines and an increased amount of energy produced per turbine, the break-even point for profitable offshore wind farms without subsidy is near.

A major issue for utilizing offshore wind farms are the uncertainties involved in the construction and operation. These appear from different sources and include uncertainties in weather and climate conditions, uncertainties in the loads on and damage of the constructions, and financial uncertainties. All of these are complicated by the long timescales involved. It is therefore of the utmost importance to quantify the uncertainties properly.

To achieve this, different aspects of uncertainty quantification are studied and combined into a framework in this thesis. The focus herein is on the case of dependent input variables which are available in the form of data rather than probability distributions. Also, a computational model to map input data to output data is assumed to be available. However, due to numerical or physical complexity, the number of available runs of this model is limited. For efficient mapping of input uncertainty to output uncertainty, the main challenge is to select suitable samples from the input data for which the model output is obtained. Additional challenges are (i) quantifying the dependencies in the data and (ii) quantifying the sensitivities of the model output with respect to the different input variables. These activities are performed before and after the runs of the computational model, respectively.

In this thesis, a method for sample selection is described which employs clustering techniques. A good representation of the input data can be achieved at relatively low computational cost, especially when the input data contains strong

dependencies. Furthermore, an efficient approach for quantifying dependencies is proposed which uses the concept of Rényi mutual information. A novel estimator is presented in case only a ranking of the dependencies is required. Computational efficiency is achieved by the application of minimum spanning trees. Sensitivity analysis can also be performed by this method, although an emulator is required to obtain predictions for more input data points. This because the number of input/output combinations is too small for an accurate analysis. Furthermore, an extension to distinguish between direct and indirect effects of input on the output is suggested.

Finally, two applications of the framework in the domain of offshore wind energy are studied. In general, the proposed framework works well and can be applied in practice.

Samenvatting

Het kwantificeren van onzekerheden met afhankelijke input-gegevens - inclusief toepassingen voor windparken op zee

Windenergie op zee is een van de alternatieve energiebronnen die in Nederland worden gebruikt om de productie van hernieuwbare energie te verhogen. Het omslagpunt waarbij windparken op zee kostendekkend worden zonder subsidie komt steeds dichterbij, door het toenemend aantal turbines en een toenemende hoeveelheid energie die per turbine geproduceerd wordt.

Een belangrijke kwestie voor het uitbaten van windparken op zee zijn de onzekerheden die samenhangen met de constructie en exploitatie. Deze zijn afkomstig uit verschillende bronnen en omvatten onzekerheden in weer- en klimaatcondities, onzekerheden in de krachten op en schade aan de constructies, en financiële onzekerheden. Het kwantificeren van deze onzekerheden wordt bemoeilijkt door de lange termijn waarop deze processen plaatsvinden en is daarom van het hoogste belang.

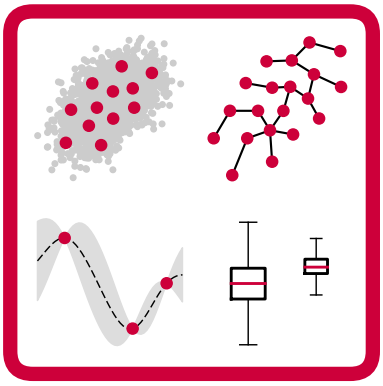
Om dit te bereiken zijn in dit proefschrift verschillende aspecten van onzekerheidskwantificatie bestudeerd en gecombineerd in een raamwerk. Het hoofdthema hierin is het geval van afhankelijke input-variabelen die beschikbaar zijn in de vorm van (meet)gegevens in plaats van kansverdelingen. Verder wordt aangenomen dat er een numeriek model beschikbaar is om input-gegevens te transformeren naar output-gegevens. Echter, het aantal simulaties dat uitgevoerd kan worden met dit model is beperkt door de numerieke of natuurkundige complexiteit. De hoofduitdaging is dan het selecteren van specifieke input-waarden op basis van de input-gegevens, die met behulp van het numerieke model worden omgezet in output-waarden. Het doel hiervan is om input-onzekerheid efficiënt te vertalen naar output-onzekerheid. Extra uitdagingen zijn (i) het kwantificeren van afhankelijkheden in de input-gegevens en (ii) het kwantificeren van sensitiviteit van de model-output ten aanzien van de verschillende input-variabelen. Deze onderdelen van het raamwerk worden respectievelijk voor en na de simu-

laties van het numerieke model uitgevoerd.

In dit proefschrift wordt een methode voor het selecteren van specifieke inputwaarden beschreven die gebruik maakt van clustertechnieken. Een goede representatie van de input-gegevens kan bereikt worden met relatief kleine numerieke inspanning, vooral wanneer de input-gegevens sterke afhankelijkheden bevatten. Ook wordt er een efficiënte methode voorgesteld op basis van het concept van wederzijdse informatie zoals gedefinieerd door Rényi. Een nieuwe schatter wordt gepresenteerd voor het geval wanneer alleen een ordening van de afhankelijkheden vereist is. Numerieke efficiëntie wordt bereikt door de toepassing van minimaal opspannende bomen. Sensitiviteitsanalyse kan ook worden uitgevoerd met deze methode, alhoewel er een emulator nodig is voor het verkrijgen van voorspellingen voor input-gegevens waarvoor geen output-gegevens bekend zijn. Dit komt doordat het aantal combinaties van input-gegevens met output-gegevens te klein is voor een nauwkeurige analyse. Verder wordt er een uitbreiding voorgesteld om onderscheid te kunnen maken tussen directe en indirecte effecten van de inputvariabelen op de output-variabelen.

Tenslotte worden twee toepassingen van het raamwerk op het gebied van windenergie op zee bestudeerd. In het algemeen werkt het raamwerk goed en kan het worden toegepast in de praktijk.

Contents



1 Introduction

Uncertainty quantification is a rapidly growing field of research which considers the uncertainties arising in all aspects of computational science. Herein, natural or artificial systems are studied by means of complex computational models. The research in this thesis is inspired by the application of uncertainty quantification to the field of offshore wind energy. We therefore introduce and illustrate the topic of uncertainty quantification by offshore wind energy.

1.1 Offshore wind energy

Offshore wind energy is an alternative power source which is capable to significantly help the Netherlands in the energy transition. The power output of offshore wind farms (OWFs) in the Netherlands has increased steadily [1] over the last few years due to two developments. First, more and more OWFs are being built and second, the turbines built are larger and produce more energy per turbine. More experience is gathered by the construction and operation of these wind farms which aids in the design of new farms. Because of these developments, the break-even point for profitable wind farms without subsidy is near. The first tenders which do not need subsidy have already been accepted [2,3] but these wind farms have not been completed yet.

One of the largest issues in the construction and operation of OWFs is formed by the uncertainties involved in it. These appear in every step in the process of planning, building and maintaining an OWF. Note OWFs have a lifespan of at least 20 years, which leads to a large time horizon in which uncertainties can appear. The uncertainties do not only appear in each step of the process, but also in each aspect. One can think of uncertainties in weather and climate conditions next to uncertainties in the loads on and damage of the constructions. Both influence the energy production, just as energy losses due to downtime of turbines or the power grid do. A third category of uncertainties are financial ones, e.g., in the cost of building and maintenance, in the discount rate (to account for the effect of the

value of money changing over time) and in the selling price of energy. Therefore, it is of the utmost importance to quantify the uncertainties properly. Especially near the break-even point, with improper quantification, this may imply that a wind farm design may be incorrectly considered as (un)profitable, leading to misguided decisions.

A computational model can be employed to quantify the uncertainties in the energy production by computing the energy production when weather conditions are given as input. This model is computationally too expensive to evaluate often and can therefore not be used to evaluate all possible weather conditions. Moreover, a model cannot capture all processes occurring in reality due to the computational cost. An extra complexity comes from the input variables being dependent. One can think of northwestern storms being more frequently observed in the Netherlands than southeastern storms, the relation between pressure and temperature or snow only being observed in combination with low temperatures.

When the uncertainties are quantified, it can be determined which of them can possibly be reduced. The research project EUROS (Excellence in Uncertainty Reduction of Offshore wind Systems) has been set up with exactly this aim, namely to develop tools to quantify and reduce uncertainties, and thereby to achieve a cost reduction of wind energy. The research which is presented in this thesis is part of this research project.

1.2 Uncertainty quantification

We introduce some mathematical notation to ease the problem description. The complex computational model is denoted by $u(\cdot)$, while $\mathbf{x}$ denotes the (multivariate) input. The output of the computational model is given by $u(\mathbf{x})$. The main setting of uncertainty quantification (UQ) is given in Figure 1.1.

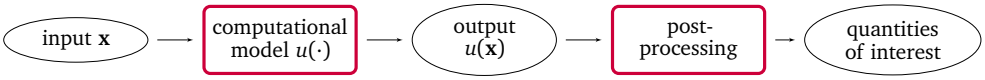


Figure 1.1: Uncertainty quantifcation.

It concerns a system consisting of a computational model together with its input and output. The output is post-processed to obtain specific quantities of interest. The computational model may be seen as a black-box, i.e., its exact functioning can be unknown to the analyst. The problem in UQ is usually the complexity of the system under consideration. For instance, the input can be high-dimensional or input data can have missing data. The process can have high complexity as well, for example a high number of parameters or long computation times, which restricts the number of evaluations. Note that the process is usually

a computational model of several physical processes, which contains uncertainty in itself as the model is typically not error-free. The combination of input and process will give the output of the system, for which one needs to quantify the uncertainties.

Due to the complexities, the number of evaluations of the computational model that one can afford is often limited. Therefore, the values of the input variables at which the computational model is evaluated, also called samples and denoted by $\{\mathbf{z}_j\}_{j=1}^{L}$ in case of L evaluations, need to be chosen carefully. This also has an effect on the subsequent parts of the uncertainty quantification, such as building an emulator or performing a sensitivity analysis. An emulator (or surrogate model) is a computationally cheap and thereby fast approximation of the computational model $u(\mathbf{x})$ [4]. In sensitivity analysis, the aim is to assess how sensitive the output uncertainty is to uncertainties in different input parameters [5]. Both activities rely on the selected input values and its model output $\{u(\mathbf{z}_j)\}_{j=1}^{L}$. Therefore, the quantities of interest also depend on the chosen samples.

When there is uncertainty in the input $\mathbf{x}$, then this uncertainty will be propagated by the model to the model output $u(\mathbf{x})$. The input $\mathbf{x}$ is usually multivariate and can be independent or dependent. Not only is dependency an important characteristic of the system, it also needs to be reflected in the uncertainty quantification. When dependencies in the input are correctly taken into account in the selection of samples, their effect will be propagated by $u(\cdot)$ to $\{u(\mathbf{z}_j)\}_{j=1}^{L}$ and taken into account in the remainder of the uncertainty quantification. The probability distribution of the input is not always known. It is also possible that the information we have about the input is in the form of data rather than probability distributions. We illustrate two characteristics of the input in Figure 1.2, where two-dimensional data is shown as input by dots; each dot represents a data point. First, the marginal distributions of the two data sets differ, as one has uniform marginals and the other one has normal marginals. Second, one of the data sets has independent variables, while the other has dependent variables. When this data would be propagated through the computational model, the cumulative distribution function (CDF) of the output $u(\mathbf{x})$ would differ. For a more extensive introduction and overview of the field of UQ, see e.g., [6–10].

The setting for UQ we consider is the following: the input is given by a large data set (i.e., many data points), of which the underlying joint and marginal probability distributions are unknown. Because of the complexity of the computational model, only a few evaluations of it can be performed. Furthermore, we especially focus on the case where the input variables are dependent. Most UQ methods available assume the input variables are independent and these methods can therefore not effectively be used on dependent data. Our goal is to develop methods for uncertainty quantification which are able to handle dependent input variables.

1.3 Framework

The need for uncertainty quantification is not limited to offshore wind energy; fluid flow problems [6, 11] such as flow around an airfoil or lid-driven cavity flow are often used as examples. Other problems which can be considered [10] are heat transfer, mechanical problems such as the simply supported beam, or ecological problems such as climate predictions [12]. General stochastic systems are studied in [13].

Our goal is to formulate and develop a framework which is able to perform UQ (including dependency analysis, uncertainty propagation and sensitivity analysis), irrespective of the underlying application. In this way, it is broadly applicable. This framework is given in Figure 1.3, where red boxes denote activities and black ovals denote input or output of these activities. It can be applied when both input data and a simulator of the process are given. We shortly explain the framework, after which we zoom in on the individual topics. The first activity is the dependency analysis in order to determine which type of sample selection is suitable. We will then focus on sample selection for dependent input data. These samples are evaluated by the simulator, which is not the focus of this thesis. As the number of evaluations of this simulator is limited because of computational cost, predictions can be made for additional input data by an emulator, which is a statistical model built to replace the simulator. Both the simulator and the emulator output can be used to obtain estimates for the quantities of interest, such as the power output of an OWF. These output values can also be used to perform a sensitivity analysis. Until now, the flow of information is from input to output and not vice-versa. During the evaluation of the framework, more and more information becomes available and it would be attractive if this can be used to improve

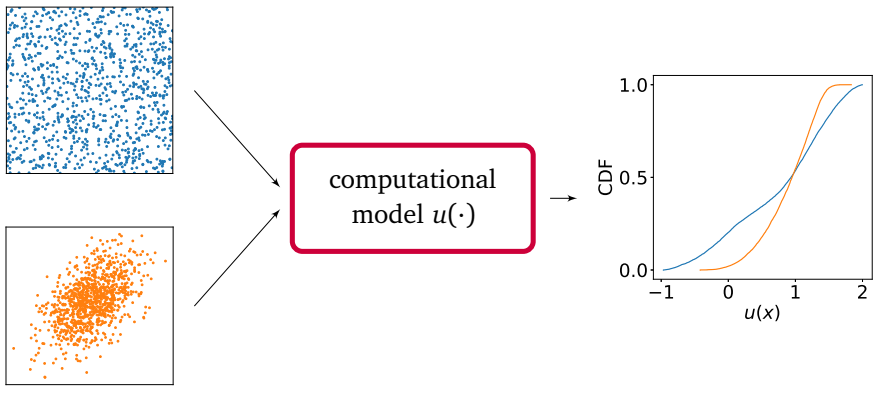


Figure 1.2: Multivariate uncertainty propagation with dependencies.

the UQ. Therefore, it is useful to add a feedback loop to the sample selection, such that not all available simulations are performed immediately, but some remain to be added if more information is available.

1.3.1 Dependency analysis

A complication for most applications nowadays is that in many existing UQ methods it is assumed that all input variables are independent. This is a rather strict assumption which can affect the results to a large extent. In the case of OWFs, high wind speeds correspond often to high wave heights. They both affect the mechanical load on the wind turbines and can thereby influence fatigue or even service life. A wind turbine which is designed to withstand an extreme wind gust or extreme wave height occurring independently of each other may fail if they occur at the same time.

For these reasons, the first step in the framework is to investigate the presence of dependencies in data sets and this is studied in Chapter 2. Our first contribution is a method to detect these dependencies fast. They may confirm general knowledge or lead to new insights. The method is based on the mathematical concept of minimum spanning trees (MSTs) and has a strong theoretical foundation in the information-theoretic concepts of entropy and divergence.

1.3.2 Sample selection

The presence of dependencies implies that most sample selection methods (for example, stochastic collocation or Latin hypercube sampling) are not suitable as they assume independent input variables. These methods often lead to tensor grids or their sparse counterparts and are vulnerable to the curse of dimensionality. If in each variable, two values are selected, and 10 variables are included, this leads already to $2^{10} = 1024$ samples. Also, because of the dependencies, this may lead to samples in regions of the domain with very low or zero density. Another issue here is that several methods not only assume independence, but also knowledge of the (marginal) input distribution(s). This is not realistic in our case, in which we consider e.g. wind speed and wave height, for which generally no explicit input distribution is known. One can think of the wind speed which is often modeled as a Weibull distribution [14], although there is no physical explanation for this.

Our second contribution is therefore a sample selection method which works for dependent data and does not require knowledge of the input probability distribution. This work is detailed in Chapter 3. The main insight is to use clustering methods established in exploratory data mining, in which working with data is much more common. These methods give sample points at which the computa-

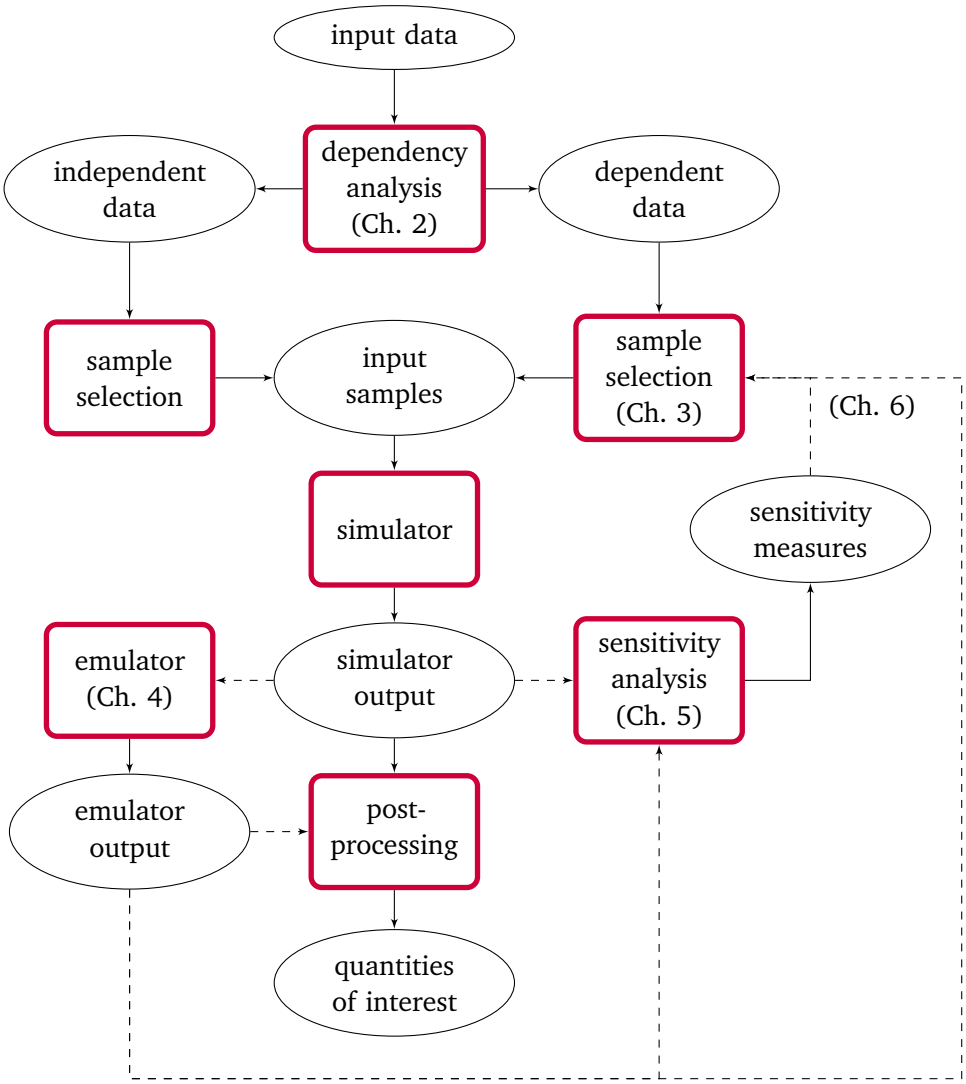


Figure 1.3: The proposed framework.

tional model is to be evaluated, together with weights of these points on which a quadrature rule can be based. A quadrature rule is an approximation of the integral of a function over a specified domain by computing a weighted sum of function values at specified locations in the domain. Here, this integral usually represents the expectation of the function $u(\mathbf{x})$ with respect to a certain input probability distribution.

1.3.3 Post-processing

Since it is hard to obtain reliable estimates for the quantities of interest from only the output of the samples, one usually applies post-processing. In a very basic setting, one constructs a quadrature rule based on the sample selection method. This quadrature rule is equivalent to integration of an emulator that consists of a piece-wise constant approximation of $u(\mathbf{x})$. In a less basic setting, one constructs a more advanced emulator which can predict the output for other input data points (other than the data points used for constructing the emulator) as well. Note that these are predictions based on a statistical model and they carry extra uncertainty with them. We study a specific type of emulators, namely Gaussian processes, in Chapter 4.

Chapter 5 treats the topic of sensitivity analysis. Often, sensitivity analysis is performed by computing sensitivity indices. We start with sensitivity indices derived previously in literature and we propose to compute them with emulated output obtained from a Gaussian process as detailed in Chapter 4. We also suggest to estimate them with minimum spanning trees, similar to the estimator used in Chapter 2, in contrast to the original estimator which uses kernel density estimation.

An open problem here is to discern between causation and correlation (dependency). In the case of an OWF, the power output can have a high sensitivity with respect to both wind speed and wave height. However, one is not inclined to say the wave height is a cause of the power output, but they are dependent through their relation with wind speed. In these cases, we want to be able to distinguish between these two types of sensitivity. Therefore, a second type of sensitivity indices has been developed as well, which supports the first type. The combination of these improvements for sensitivity analysis is our third main contribution.

We have now reached the end of the information flow in Figure 1.3 and are going to move back in the direction of information. The information delivered in the sensitivity analysis can be used if extra samples for the same process are later available. One can then add the remaining samples in such a way that they are most beneficial given the current knowledge enclosed in the emulator and the sensitivity analysis.

We propose a two-stage sampling method in Chapter 6. The first stage can

be chosen at will, while the second stage is constructed in a special way. It is based on clustering combined with some properties of the emulator described in Chapter 4. An advantage of this method is the decrease of the prediction variance after updating the Gaussian process, which makes the estimates obtained with the emulator more precise.

1.3.4 Test cases

Chapters 2 to 6 give detailed information about the steps in the framework illustrated in Figure 1.3. Chapters 7 and 8 contain test cases to illustrate the use of the framework or parts thereof.

The complete process will be studied in Chapter 7 for a test case regarding input data from Meteomast IJmuiden [15]. This is a meteorological mast located in the North Sea, approximately 75km west of the IJmuiden coast (the Netherlands). Several atmospheric conditions such as wind speed and direction, air pressure and temperature have been measured at several heights, together with sea conditions. The data has been post-processed and is publicly available online. This data is combined with a computational model of Horns Rev I. This offshore wind farm in Denmark is well known in the wind energy community [16–19] for its wind turbine wakes. Despite the complexity of the model, it can be evaluated fast, which makes it suitable to compute the accuracy of the studied methods.

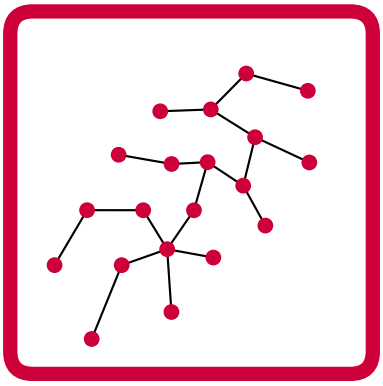
In Chapter 8, we will take a slightly different approach. Although the same computational model is used, the data differs and is obtained from the Royal Netherlands Meteorological Institute (KNMI). Here, the quantity of interest is the levelized cost of energy and we investigate how this quantity depends on the weather data used to perform the energy calculations.

Concluding remarks and suggestions for future research are given in Chapter 9.

1.4 Setting and implementation

The setting of this thesis is to be understandable for a more general audience than mathematicians only. Therefore, we indicate sections which are theoretic or very technical of nature and which are not required for a general understanding of the treated topics by a star ($\star$).

Most chapters in this thesis contain extensive numerical results. These are not obtained by the use of computing clusters or supercomputer facilities, but by a single laptop or desktop computer instead. Therefore, no computational hurdles are raised against the implementation and use of the proposed methods. The developed code together with an example is published online with the digital object identifier 10.5281/zenodo.3235924.



2 Dependency analysis

The goal of this chapter is to develop a method to detect dependencies in data. These may confirm general knowledge or lead to new insights. Furthermore, this serves as a precursor for the sensitivity analysis studied in Chapter 5.

2.1 Introduction

The question to what extent random variables are dependent emerges in various places. In uncertainty quantification (UQ), the uncertainties in simulation models of complex systems are explored, for example by assessing what the probability distribution of the simulation output is given the probability distribution(s) of various uncertain model input variables [6, 9, 10]. Many methods in UQ are designed for situations where the random input variables are mutually independent, hence the (in)dependence of these inputs is of obvious importance.

A topic in UQ that is particularly relevant for this study is sensitivity analysis, where it is investigated which random inputs induce the largest uncertainties in the simulation output [5, 20, 21]. Sensitivity analysis is closely related to the question how strongly dependent the output is on the individual input variables. By quantifying these dependencies, one can order the inputs by the extent to which the output is dependent on them (from strongly to weakly dependent), providing relevant information for sensitivity analysis.

In this chapter we consider quantification of dependencies between random variables in situations where (i) the distribution of the random variables is unknown and only a given, finite sample from the joint distribution is available, and where (ii) the dependencies can be nonlinear, in contrast to dependencies in multivariate Gaussian distributions. It is well-known that information theory forms a suitable framework to deal with non-Gaussian distributions and nonlinear depen-

Mainly based on A. Eggels and D. Crommelin, "Quantifying Data Dependencies with Rényi Mutual Information and Minimum Spanning Trees," *Entropy*, vol. 21, no. 2, article no. 100, 2019.

dencies [22]. However, information-theoretic quantities such as Shannon entropy and Kullback-Leibler (KL) divergence are formulated in terms of probability distributions or probability density functions, precluding their exact computation if the distribution is unknown.

As an alternative, one can estimate entropy (and related quantities) from sample data. Such estimation is nontrivial however. A typical approach is to use a plug-in estimator for the probability density (e.g. by binning or by kernel density estimation) to compute (an estimate of) the entropy. This approach faces problems relating to the difficulties of estimating densities, see e.g. [23] for a discussion. To bypass these problems, Hero et al. [23–25] proposed an appealing and elegant method for estimating entropies from data by computing minimum spanning trees (MSTs) of the data and using the length of the MST in an estimator for Rényi entropy. This method does not require estimation of probability densities. In this chapter, we employ the method from [23–25] for quantifying dependencies, by using the length of the MST as a measure of dependence between random variables.

In principle, MSTs can be computed exactly, with algorithms due to Kruskal [26] and Prim [27]. However, these algorithms become expensive in case of large data sets. Therefore we test several approximate methods to compute the MST length with strongly reduced computational cost.

Our main contribution is to accelerate the method in [23] by computing an approximate rather than an exact MST. In the era of data science, it is expected that large data sets will appear where a speedup of the computations is desirable. The question herein is whether the approximation is accurate enough to replace the costly but exact MST. Our second contribution is that we show that it is not necessary to compute the Rényi divergence to quantify dependence for the purpose of ranking dependency strength between different random variables. An estimator derived from it is equally informative yet easier to compute.

Although the present chapter is concerned primarily with dependency analysis rather than sensitivity analysis, the latter is an important motivation for the work reported here. We will employ the tools developed here for sensitivity analysis in Chapter 5, and note that the relevance of entropy estimation for sensitivity analysis was discussed before in e.g. [28, 29]. These studies rely, however, on density estimation to compute entropy. The MST approach to entropy estimation from [23–25] has to our knowledge not been considered before in the context of dependency analysis and sensitivity analysis.

This chapter is organized as follows: In Section 2.2 we summarize some theory regarding entropy and we relate Rényi entropy to Rényi divergence. Furthermore, we discuss the estimation of Rényi entropy with the MST. Next, in Section 2.3 the approximation methods for computing the MST are discussed. Section 2.4

consists of validating the proposed estimator and determining the consistency and robustness of the approximations to it. In Section 2.5 we apply the proposed methods to several test cases. Section 2.6 concludes the present chapter.

2.2 Dependence and entropy ⋆

We summarize a few basic definitions and concepts in Sections 2.2.1 and 2.2.2. Section 2.2.3 describes the estimator of Rényi entropy as developed in [23]. This paves the way for defining our proposed estimator for dependency between random variables in Section 2.2.4. This estimator is related to, but not equal to the Rényi mutual information. Section 2.2.5 concerns a proof-of-concept.

2.2.1 Rényi entropy and divergence

Rényi entropy is a generalization of the Shannon entropy. The latter, also known as differential entropy in the continuous case, is defined as (see e.g. [22])

$$H(X) = -\int_{\Omega} p(x)\log(p(x))\mathrm{d}x, \tag{2.1}$$

where $p(x)$ is the probability density function of the continuous random variable X, and Ω is the domain of X. The generalization by Rényi reads

$$H_\alpha(X) = \frac{1}{1-\alpha}\log\left(\int_{\Omega} (p(x))^\alpha \,\mathrm{d}x\right), \tag{2.2}$$

for $\alpha \in (0,\infty)$, see [22]. In the limit $\alpha \to 1$, the Rényi entropy (2.2), sometimes referred to as α-entropy, converges to the differential entropy (2.1).

Related to the Rényi entropy (2.2) is the Rényi divergence [30] of a probability density function $f(x)$ with respect to another probability density function $g(x)$:

$$D_\alpha(f,g) = \frac{1}{\alpha-1}\log\left(\int_{\Omega}\left(\frac{f(x)}{g(x)}\right)^\alpha g(x)\mathrm{d}x\right). \tag{2.3}$$

This divergence is well-defined if g dominates f (i.e., $f(x^*) = 0$ for all x^* where $g(x^*) = 0$). It converges to the Kullback-Leibler (KL) divergence $D_{\mathrm{KL}}(f,g) = \int f(x)\log(f(x)/g(x))\mathrm{d}x$ for $\alpha \to 1$. If $f = g$ almost everywhere we have $D_\alpha(f,g) = 0$, otherwise $D_\alpha(f,g) > 0$.

2.2.2 Entropy as measure of dependence

It is well-known that the dependence between two random variables Y and Z can be characterized by their mutual information, i.e. the difference between the sum of the Shannon entropies of Y and Z individually and the Shannon entropy of (Y,Z) jointly.

$$I(Y,Z) = H(Y) + H(Z) - H(Y,Z),$$

where $H(Y,Z)$ is the joint entropy of Y and Z (see [22]). Shannon entropy can also be defined by the KL divergence of the joint density $g_{yz}(y,z)$ with respect to $\tilde{g}_{yz}(y,z)$, the product of the marginal densities $g_y(y)$ and $g_z(z)$ (where $g_y(y) = \int g_{yz}(y,z)\mathrm{d}z$ and $g_z(z) = \int g_{yz}(y,z)\mathrm{d}y$).

We can generalize this information-theoretic characterization of dependence by using Rényi rather than KL divergence. Thus, we quantify dependence by $D_\alpha(g, \tilde{g})$ (see (2.3)), the Rényi divergence of $g_{yz}(y,z)$ with respect to $\tilde{g}_{yz}(y,z)$. This can also be used to define the Rényi mutual information [31, 32] as

$$I_\alpha(Y,Z) = \frac{1}{\alpha-1}\log\left(\left(\int_Y\int_Z\left(\frac{g_{yz}(y,z)}{g_y(y)g_z(z)}\right)^\alpha g_y(y)g_z(z)\mathrm{d}z\mathrm{d}y\right)\right).$$

We note that in the limit $\alpha \to 1$, $I_\alpha \to I$. It is easy to verify that by construction, $g_y(y)g_z(z)$ dominates $g_{yz}(y,z)$, hence the Rényi and KL divergences are well-defined. As was shown in [33], the divergence $D_\alpha(g, \tilde{g})$ is equivalent to the Rényi entropy of a different density, resulting from a coordinate transformation. Let $(y',z') = G(y,z)$ where $G : \mathbb{R}^2 \mapsto \mathbb{R}^2$ is the Rosenblatt transformation [34, 35] induced by the product density $\tilde{g}$ (i.e., the transformation that uniformizes g). The joint density induced by this transformation is $h(y',z')$, given by

$$h(y',z') = \frac{g_{yz}(G^{-1}(y',z'))}{\tilde{g}_{yz}(G^{-1}(y',z'))}, \tag{2.4}$$

and as a result,

$$\begin{aligned} D_\alpha(g, \tilde{g}) &= \frac{1}{\alpha-1}\log\left(\int\left(\frac{g_{yz}(y,z)}{\tilde{g}_{yz}(y,z)}\right)^\alpha \tilde{g}_{yz}(y,z)\,\mathrm{d}y\mathrm{d}z\right), \\ &= \frac{1}{\alpha-1}\log\left(\int h^\alpha(y',z')\,\mathrm{d}y'\mathrm{d}z'\right), \\ &= -H_\alpha(h). \end{aligned} \tag{2.5}$$

Thus, $D_\alpha(g, \tilde{g})$ equals $-H_\alpha(h)$, the Rényi entropy of the induced joint density h given in (2.4), up to a minus sign. This makes sense because a stronger dependence is coherent with a smaller (more negative) $H_\alpha(h)$ and a larger $D_\alpha(g, \tilde{g})$. In addition, we note that (2.4) is similar to the copula density (see also [31]).

If the density $g_{yz}(y,z)$ (and hence also $\tilde{g}_{yz}(y,z)$) is unknown and only a sample of its probability distribution is available, we can (approximately) carry out the coordinate transformation G induced by $\tilde{g}$ without having to estimate g or $\tilde{g}$ itself. Instead we apply the rank transform [36, 37], together with centering, to the data. The marginals of the rank-transformed data are discrete representations of the uniform distribution $U[0,1]$.

In summary, if we are given a sample of the random variables (Y,Z) with joint probability density $g_{yz}(y,z)$, we can estimate the Rényi divergence $I_\alpha(g,\tilde{g})$ by first applying the rank-transform to the sample data, and then estimating the Rényi entropy of the transformed data $H_\alpha(h)$.

Before turning to the estimation of the Rényi entropy, we conclude this subsection by pointing out the connection to another measure of divergence or dissimilarity between probability densities, namely the Hellinger distance, and by commenting on the choice of α. The Hellinger distance [38] can be computed from $D_{1/2}$ by easy transformations. The advantages of using $D_{1/2}$ are that it is a distance metric [23, 39] and that it is optimal in the context of classification problems [39].

2.2.3 Estimator of the Rényi entropy

It is not straightforward how to estimate the entropy of a given data set. As already discussed in the introduction of this chapter, one approach is to estimate the probability distribution or density underlying the data set, and compute the entropy from this estimated density. Both parametric and nonparametric methods (e.g. kernel-density estimation, binning) can be used to estimate the distribution, however the results are quite sensitive to the details of the method used [40]. See also the work in [23] where the difficulties of entropy estimation by means of plug-in density estimators are summarized. Noshad et al. [41] circumvented the plug-in density and proposed a direct estimation method based on a graph-theoretical interpretation. Very recently, Moon et al. [32] derived mean-squared error convergence rates of kernel density-based plug-in estimators of mutual information and proposed ensemble estimators that achieve the parametric rate, although there are several restrictions on the densities and the kernel used.

To circumvent the need for density estimation, we employ a different approach to estimate entropy in this study, one in which the sample data is used directly. This approach is due to Hero & Michel [23–25], who proposed a direct method to estimate the Rényi entropy, based on constructing minimal graphs spanning the data points. This approach can be used to estimate $H_\alpha(X)$ with $0 < \alpha < 1$, hence it does not apply to estimation of the Shannon entropy. This motivates our focus on the Rényi entropy here.

Let X_N denote a sample of the multivariate random variable X with sample

size N. The dimension of the domain of X is d. The estimator proposed in [23–25] is

$$\hat{H}_\alpha(X_N) = \frac{1}{1-\alpha}\left(\log\left(\frac{L_\gamma(X_N)}{N^\alpha}\right) - \log\beta_{L,\gamma}\right) = \frac{1}{1-\alpha}\log\left(\frac{L_\gamma(X_N)}{\beta_{L,\gamma}N^\alpha}\right), \tag{2.6}$$

where $\alpha = (d-\gamma)/d$ and $\beta_{L,\gamma}$ is a constant only depending on γ and the definition of L_γ. The functional $L_\gamma(X_N)$ is defined as

$$L_\gamma(X_N) = \min_{T(X_N)} \sum_{e\in T(X_N)} |e|^\gamma, \tag{2.7}$$

where $T(X_N)$ is the set of spanning trees on X_N and e denotes an edge. The parameter γ can be freely chosen within the interval $(0, d)$, with $d \geq 2$. The choice of γ determines α, and $\alpha \in (0, 1)$ because $\gamma \in (0, d)$. The estimator (2.6) is asymptotically unbiased and in fact strongly consistent for $\alpha \in (0, 1)$ [33]. If γ is chosen to be 1 and $|e|$ denotes the Euclidean distance between data points in X_N, then $\gamma(X_N)$ describes the length of the MST on the data set. In this chapter we focus on $d = 2$, but we note that (2.6) applies to situations with $d > 2$ as well. With $d = 2$ and $\gamma = 1$ we have $\alpha = 1/2$, a value for α whose theoretical motivation was discussed earlier.

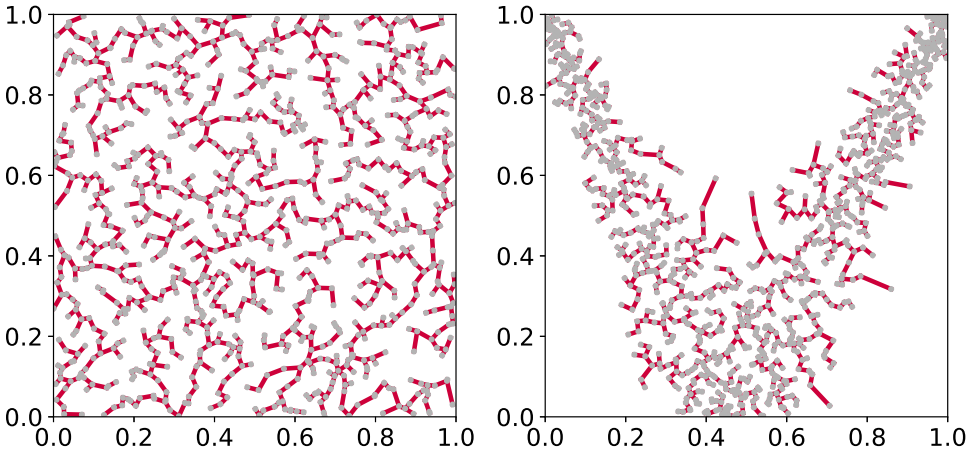
We give an illustration of the MST on data in the unit square, i.e. $d = 2$ (we recall that data sampled from distributions on other domains will be mapped to the unit (hyper)square by the rank transform). In case of independence the MST will approximately cover the full domain. In case of dependence, the density is manifestly nonuniform. As a result, the average edge length decreases and so does the total length of the MST. This is illustrated in Figure 2.1. A shorter MST length (with N constant) implies lower entropy, cf. (2.6), and therefore stronger dependence. Note that there is no assumption on the shape or structure of the dependence.

2.2.4 Quantifier of dependence

The value of the constant $\beta_{L,\gamma}$ in (2.6) is unknown, but it is defined as

$$\beta_{L,\gamma} = \lim_{N\to\infty} L_\gamma(X_N)/N^\alpha, \tag{2.8}$$

for X_N a sample of size N from the bivariate uniform distribution. Therefore, it does not depend on the distribution of X and can be regarded as a constant offset or bias correction. When comparing the entropies of two random variables $X^{(1)}$ and $X^{(2)}$ with $\dim(X^{(1)}) = \dim(X^{(2)}) = d$, the difference $\hat{H}_\alpha(X_N^{(1)}) - \hat{H}_\alpha(X_N^{(2)})$ does not depend on $\beta_{L,\gamma}$ provided the same type of spanning tree is used (i.e., same functional L and γ).



(a) Independent data. (b) Dependent data.

Figure 2.1: Illustration of the MST for two data sets on the unit square. One data set is sampled from a bivariate independent distribution (left panel), the other from a strongly nonlinear dependent distribution (right panel).

Therefore we propose to use the following quantity to measure (or quantify) dependence:

$$H^*_\alpha(Z_N) = \log\left(\frac{L_\gamma(Z_N)}{N^\alpha}\right), \tag{2.9}$$

where Z_N is a data set consisting of the rank-transform of X, containing N data points. We set $\alpha = 1/2$, as discussed in the previous section. We focus here on the case $d = 2$ so that $\gamma = 1$. The advantage of this estimator is that, in the case where multiple dependencies are compared, the value of $\beta_{L,\gamma}$ is not necessary to obtain an ordering of them, and, therefore, this value does not need to be estimated.

Besides obtaining an ordering in terms of the dependency strength, we can use (2.9) to construct a reference level for distinguishing between independent and dependent variables. This construction consists of computing $H^*_\alpha(Z_N)$ on n_r different data sets of rank-transformed bivariate uniformly distributed data. These data sets are generated independently of each other, and, within each data set, the two variables are independent as well (as they have a bivariate uniform distribution). This is done by sampling $2N$ values from the one-dimensional uniform distribution and reshaping them into an $N \times 2$ data set. From this we obtain an empirical distribution, based on n_r samples, for $H^*_\alpha(Z_N)$ for independent variables (we recall that data from independent variables always leads to data uniformly distributed on a grid after the rank-transform). This leads to the following statistical test for dependence. The null hypothesis is that the random variables are inde-

pendent, which implies this null hypothesis is rejected if $H_\alpha^*(Z_N)$ is significantly smaller (more negative) than could be expected if the variables were independent. Thus, the null hypothesis is rejected if

$$H_\alpha^*(Z_N) \leq \eta,$$

in which η is the 0.01 or 0.05 quantile of the empirical distribution for $H_\alpha^*(Z_N)$ for independent variables. In the remainder of this text, we will write $H_\alpha^*(Z)$ instead of $H_\alpha^*(Z_N)$ to simplify the notation. It will be clear from the context whether data is involved or not.

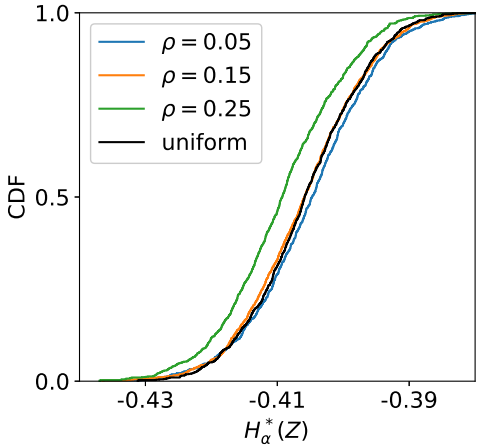
2.2.5 Proof of concept

First, we compute the quantifier of dependence (2.9) on multiple data sets with varying distributions to analyze its behavior. Then, its convergence and robustness for increasing values of N are studied. For the distributions considered in this section, it is possible to compute the Rényi entropy with high accuracy for the scaled (i.e., rank-transformed) distribution such that a comparison can be made between the behavior of (2.9) and (2.2).

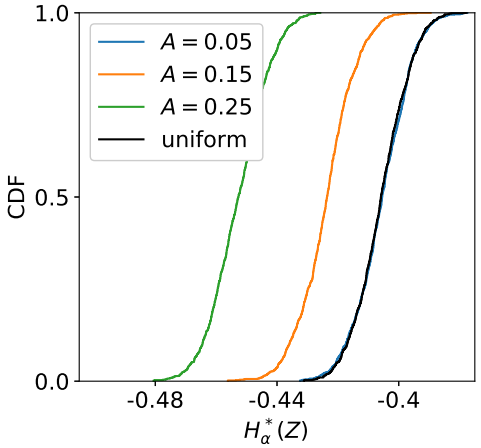
We use data sets sampled from the following distributions: (i) a bivariate uniform distribution, (ii) a standard normal distribution with varying correlation coefficient ρ, (iii) a constant density on a region within the unit square with area $1-A$, in which the data is focused in the lower left corner (corner distribution), and (iv) a constant density on a region within the unit square with area $1-A$, but in which the data is focused on a symmetry axis (line distribution). For the latter two cases, the region includes values near the minimum and maximum of both variables, such that the ranges of the region for varying A stay the same. These distributions are also referred to as shape distributions. For each of the four distributions, $r = 10^3$ data sets are generated with $N = 10^3$ data points in every data set. ρ and A are varied in steps of 0.10 from 0.05 to 0.95.

In Figure 2.2, the empirical cumulative distribution function (CDF) of $H_\alpha^*(Z)$ is plotted for both the normal distribution (case (ii)) and the two shape distributions (cases (iii) and (iv)), for different small values of ρ and A. More precisely, for each data set sampled from one of these distributions, we first apply the rank-transform and then evaluate (2.9) for the rank-transformed data. The empirical CDF for the uniform distribution is included in black for comparison. If $\rho = 0$ or $A = 0$, there is no dependence anymore and the empirical CDF obtained from the normal distribution or shape distributions coincides with the empirical CDF from the bivariate uniform distribution (modulo differences due to finite sample size, $N < \infty$).

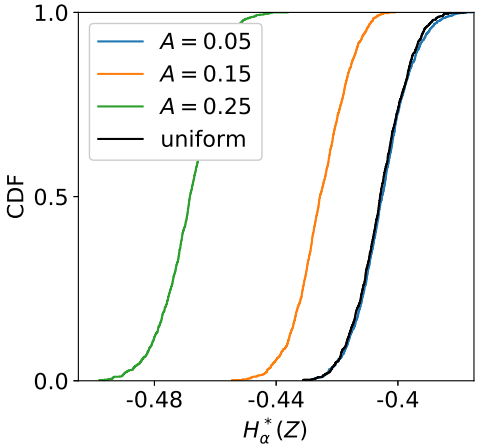
In Figure 2.2, it can be seen that for the shape distributions, the quantifier of dependence is more distinctive for small values of A than it is for the normal



(a) Normal distribution.



(b) Corner distribution.

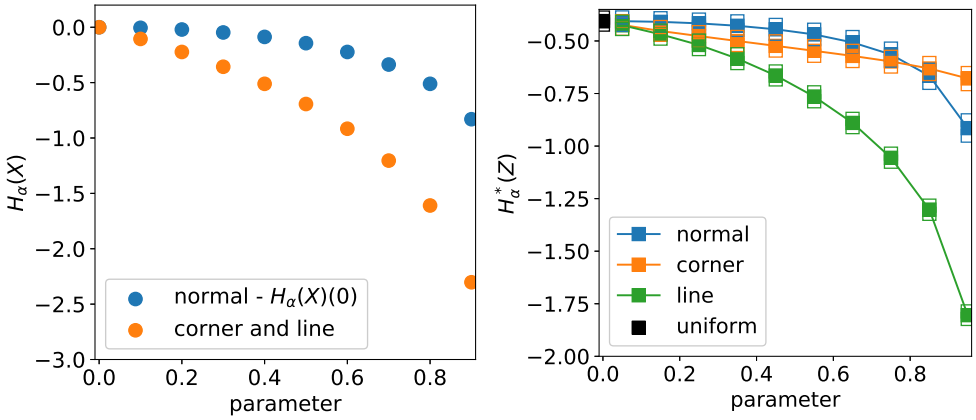


(c) Line distribution.

Figure 2.2: Empirical distributions of $H^*_\alpha(Z)$ for data from the normal distribution and the two shape distributions, for small ρ and A.

distribution with small ρ. Hence, weak dependencies in the shape distributions can more easily be detected than in the normal distribution. The behavior of $H_\alpha(X)$ is given in Figure 2.3a. To demonstrate the behavior of $H^*_\alpha(Z)$ for the full range of values of ρ and A, we plot the mean together with the empirical 95% confidence intervals in Figure 2.3b. In the case of $\rho = A = 0$, the uniform (independent) distribution is recovered, indicated by a different color.

From Figure 2.3a, it can be seen why the differences between the CDFs are small in case of the normal distribution: for the normal distribution, the entropy



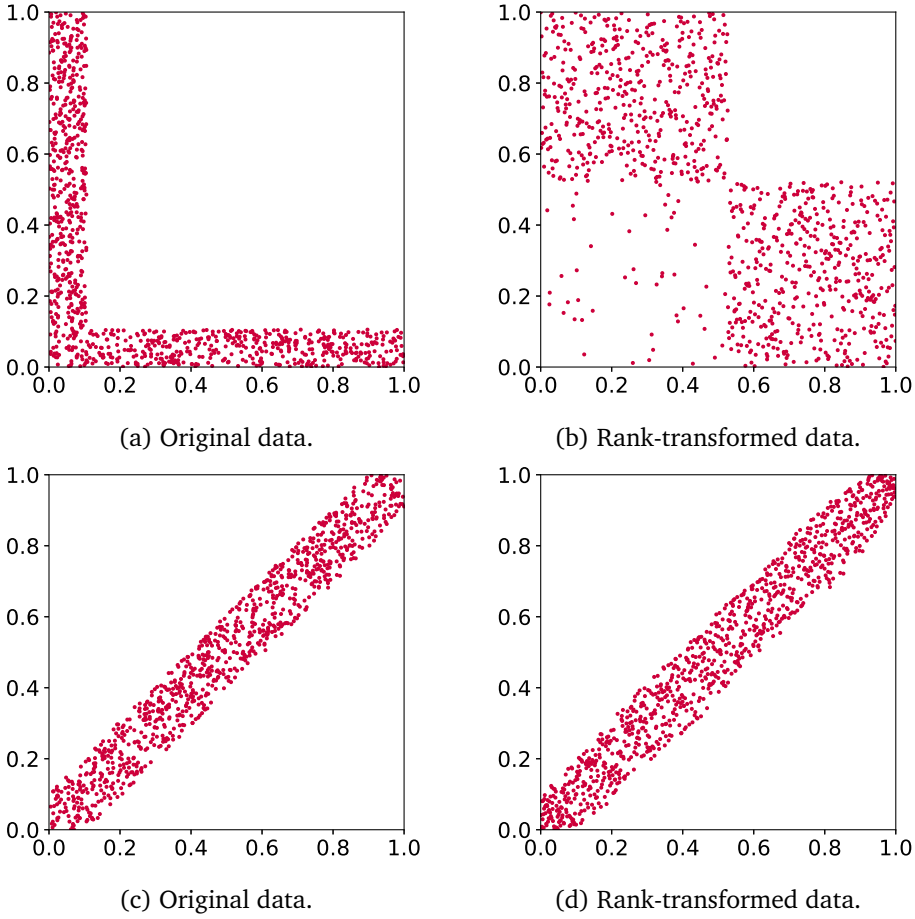
(a) Rényi entropy.

(b) Quantifier of dependence.

Figure 2.3: Comparison of the Rényi entropy (left) and quantifier of dependence (right) for the normal distribution and shape distributions, with varying parameters (ρ and A). The Rényi entropy is computed exactly using (2.2) without transforming the data. For visualization purposes, the values for the normal distribution have been translated by its value for $\rho = 0$, which is $2\log(2\sqrt{2\pi})$. The quantifier (2.9) is evaluated using data that is sampled from the distribution and then rank-transformed. We show the mean and 95% confidence intervals of $H^*_\alpha(X)$. Note that the numerical values of the entropy and $H^*_\alpha(X)$ are not supposed to coincide, cf. (2.6) and (2.9).

is nearly flat as a function of ρ, for small ρ values. Figure 2.3b shows the rank-transform has a more severe effect on the corner than on the line distribution, since the estimates for the corner distribution decrease more slowly. The distortion of the shape of the graph for the corner distribution between Figure 2.3a and 2.3b is caused by the distortion of the distances between data points after the rank-transform. Note that both shape distributions have the same Rényi entropy if the rank-transform is not applied. The effect of the rank-transform in this case can be seen in Figure 2.4. The closer A is to 1, the more the data falls in the two boxes for the corner distribution, due to the combination of skewness and discontinuity. This does not hold for the line distribution. In the limit of $A \to 1$ and $N \to \infty$, the entropy of the rank-transformed corner distribution goes to $-\log(2)$, while it diverges to $-\infty$ for the rank-transformed line distribution. Hence, it is consistent that $H^*_\alpha(Z)$ does not diverge to $-\infty$ for the corner distribution. We note that the shape distributions are quite artificial and have discontinuous density, while data sets in practice are usually samples from a distribution with a smooth density and a less artificial shape.

We conclude this section by investigating the effect of varying the size of the data set, N. We compute the empirical CDF of $H^*_\alpha(Z)$ using $N \in \{10, 10^2, 10^3\}$.



(a) Original data.

(b) Rank-transformed data.

(c) Original data.

(d) Rank-transformed data.

Figure 2.4: Example of the data and its rank-transform for the shape distribution with $A = 0.8$. Top row: corner distribution, bottom row: line distribution.

For this computation we keep $n_r = 10^3$. The resulting empirical CDFs are shown in Figure 2.5. It can be seen the distribution becomes narrower with increasing N, as expected. Thus, larger N makes the comparison of estimates easier due to the smaller confidence intervals involved. One can see from this figure how $\beta_{L,\gamma}$ is defined as the limit for $N \to \infty$. Therefore, when one estimates $\beta_{L,\gamma}$ from a finite data set, one overestimates its value. This can be seen by the value for $H^*_\alpha(Z)$ at which the CDF equals 0.5: this value moves to the left for increasing N.

We note that for higher values of N, the computations become expensive due to the computation of the edge lengths before computing the MST. The order of this operation is $O(N^2)$. Kruskal's method for computing MSTs [26] runs in $O(N^2 \log(N))$ time, but the steps are faster than the ones needed for computing

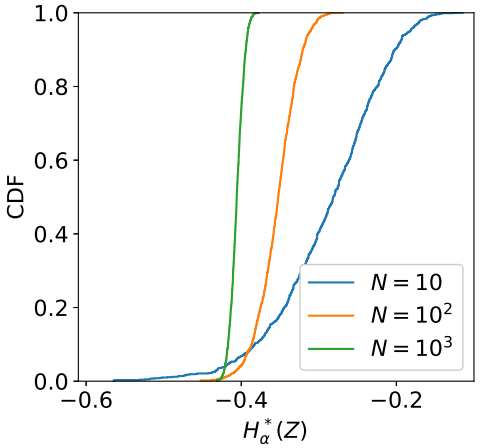


Figure 2.5: Empirical CDF of $H_\alpha^*(Z)$ for the bivariate uniform distribution. The CDF becomes narrower for larger data sets (increasing N).

the edge lengths. Prim's method for computing MSTs [27] can be faster ($O(N^2 + N\log(N))$), but the implementation is more involved. The high cost of computing the exact MST motivates us to investigate approximate algorithms in the next section.

2.3 Approximation methods

For large data sets (i.e., large N), the computational cost of evaluating the estimator (2.9) becomes prohibitively high, due to the computational complexity of constructing the MST. In this section we discuss methods to reduce the computational cost by approximating the value of (2.9) evaluated on a large data set. We consider three types of approximation: first, MSTs can be computed on multiple subsets of the data. The second type aggregates data points into clusters, such that only one MST calculation is performed on the cluster centers. The third type clusters the data points, constructs MSTs on each cluster and combines them in a smart way [42].

2.3.1 Sampling-based MST

In this method, the data set is split in K subsets and $H_\alpha^*(Z)$ is computed according to (2.9) on each of the subsets. Thus, K estimates of $H_\alpha^*(Z)$ can be obtained. The new estimate is computed from the mean of the MST lengths, while their variance can serve as a measure for the quality of the new estimate. The number

K is chosen by the user, although it represents a trade-off between accuracy and computation time (since both decrease with increasing K).

The splitting can be done in various ways, of which random and stratified are the most straightforward ones. In the random splitting, data points are allocated randomly to one of the K subsets, which all have size N/K (rounding is neglected). In the proportional splitting, $k = N/K$ clusters are generated with the k-means method [43, 44] and these are considered the strata. The data points in each stratum are then proportionally allocated to the K subsets. An example with stratified splitting, $N = 10^3$ and $K = 10$ can be found in Figure 2.6.

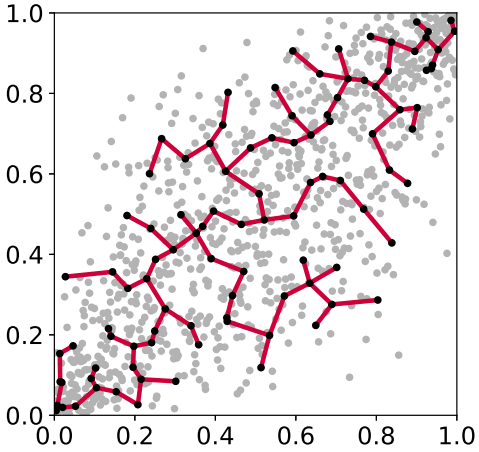


Figure 2.6: Example with stratified splitting.

2.3.2 Cluster-based MST

Another way of reducing the computational burden would be to cluster the data in a large number of clusters and compute the MST on the cluster centers. In this case, the clustering method can be chosen, as well as the number of clusters. To be consistent with the previous method, we define here the number of clusters to be $k = N/K$, such that the number of points in the MST is similar to the previous method. Furthermore, it is possible to include the size (weight) of the clusters as well. Two different clustering methods are investigated: k-means [44] and principal component analysis-based (PCA-based) clustering (see also Sections 3.3.1 and 3.3.2). The concept of clustering is explained in more detail in Section 3.3. The construction of the MST is performed both without and with weighting. The weighting is harmonic, which implies that in the construction of the MST, the edge

length between data points i and j, $|e_{ij}|$, is replaced by

$$W_{ij}|e_{ij}|, \quad W_{ij} = \frac{2k}{\frac{1}{w_i} + \frac{1}{w_j}},$$

where k is the number of clusters and w_i, w_j are the weights of the clusters, computed by the fraction of data points in that cluster. An example with these weights (again with $N = 10^3$ and $K = 10$) is given in Figure 2.7, in which the line width indicates W_{ij}. One can see this method behaves according to the principle of least resistance: small clusters serve as a hub to connect the larger clusters.

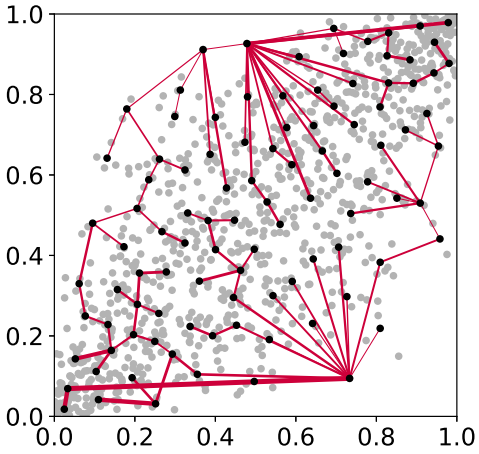


Figure 2.7: Example with k-means clustering and a weighted MST.

2.3.3 Multilevel MST

This method has been proposed by Zhong et al. [42] and is also called FMST (fast MST). It is based on the idea that to find a neighbor of a data point, it is not necessary to consider the complete data set of size N. In the FMST method, the data set is partitioned in $\sqrt{N}$ clusters via the k-means clustering method, and MSTs are constructed on each of the clusters. More details on this clustering can be found in Section 3.3.1. In a next step, an MST is constructed on the cluster centers to determine which clusters get connected at their boundaries. Between these clusters, the shortest connecting edges are computed. Because this is a heuristic procedure, the complete process is repeated with the midpoints of the edges which connect different MSTs being the initial cluster centers for the k-means method. The resulting two MSTs are merged into a new graph and its MST is computed as final outcome of the method. [42] reports good results with

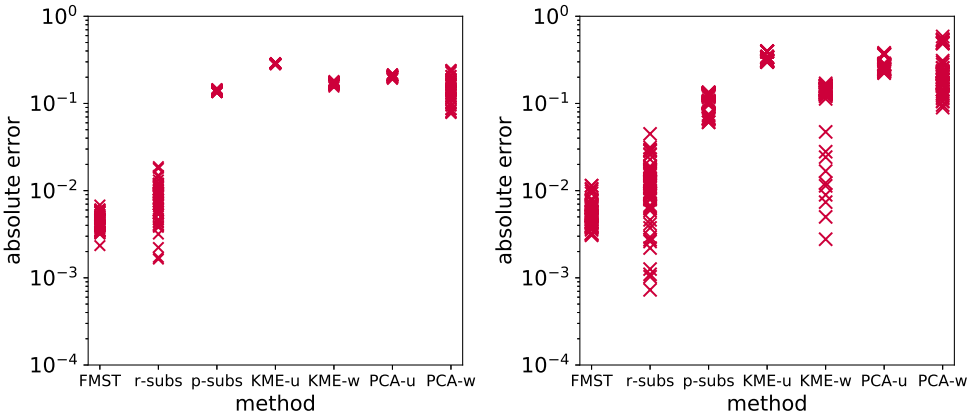
this method, which has a computational complexity of $O(N^{1.5})$. Errors occur only if points that should be connected end up in different clusters twice and are not chosen as connecting edge, which does not occur often and has only a minor effect. For high-dimensional data sets, erroneous edges are included more often, but the effect thereof is smaller. Both are due to the curse of dimensionality.

2.3.4 Comparison

The accuracy, robustness and computational time of the methods explained before are tested on $r = 50$ bivariate uniform data sets with $N = 10^4$ and $r = 60$ bivariate strongly dependent data sets, also with $N = 10^4$. The strongly dependent data sets are chosen as projections of the data set used in Section 2.5.1. Hence, the data set exists of combinations of x, y, z and $I(x, y, z)$. Since this leads to 6 projections per data set, we choose $r = 60$ in this case. In these cases, it is computationally expensive but still feasible to compute the full (exact) MST, and we do so to be able to compare results from the three approximate methods with the method based on the full MST. For the approximate methods, we choose the parameter K to be 10. The results can be found in Figure 2.8. We mention here the effect of the implementation of k-means, as the choice of initialization can affect the results. We use the k-means++ initialization algorithm, repeat the clustering 10 times from different initializations and select the result with the best clustering criterion. The error estimate is computed as the absolute difference between the approximated and exact value of $H^*_\alpha(Z)$. The FMST method has consistently small errors as compared to other methods. In our opinion, it is important that the approximation is accurate and robust, i.e., has little variation in its error over multiple runs of data with the same distribution. Furthermore, it should perform well for both dependent and independent data. Therefore, we propose to use the FMST method to compute approximations to the MST in case where the data set is large (say, $N \geq 10^4$).

2.4 Validation of the proposed FMST estimator ⋆

In this section, we further investigate using the FMST method with the estimator (2.9). First, the effect of the size of the data set N is investigated together with the robustness of the FMST estimator for relatively small N. Then, the FMST estimator for dependence is tested for behavior and consistency using data sets sampled from the three distributions considered before (uniform, normal and shape distributions). Furthermore, we investigate the behavior of the dependence measure obtained with the MST method and the FMST method to the exact value in a separate simulation study in Section 2.4.1.



(a) Independent data.

(b) Dependent data.

Figure 2.8: Error estimates for different approximation methods. The figure on the left is for independent data, while the figure on the right is made with dependent data. From left to right, the methods are: FMST, random sampling from subsets-based MST, stratified sampling from subsets-based MST, k-means cluster-based MST (unweighted and weighted) and PCA-based cluster-based MST (unweighted and weighted).

It is straightforward that approximations to the MST using the correct edge distances overestimate the length of the MST. Therefore, we compare the empirical distributions of $H_\alpha^*(Z)$ based on the MST (Figure 2.5) to the ones based on the FMST in Figure 2.9. The number of repetitions is $r = 10^3$ for $N = 10^2$, 10^3 and $r = 10^2$ for $N = 10^4$ and $N = 10^5$. It can be seen that the distributions are different, but only shifted. Furthermore, the width of the distribution decreases to almost zero for $N = 10^4$ and $N = 10^5$. This means the estimator is robust to sampling effects. Note that there is still a difference between the distributions for $N = 10^4$ and $N = 10^5$. This is due to the bias caused by approximating the MST, which reduces for increasing N.

We repeated the experiment from Section 2.2.5, but now with the estimator based on the FMST method to evaluate the effect of the FMST approximation on different data distributions. The results are in Figure 2.10. Again, the behavior of the estimator based on the FMST is the same as for the one based on the MST, but only shifted upwards. This bias is very small for the line distribution. In these tests, $N = 10^3$ in order to compare their results to the previous results.

Based on these results, we conclude that the FMST estimator is a good and robust approximation of the MST estimator. It has a positive bias, but for the purpose of comparing and ranking strength of dependencies as quantified by the FMST estimator, this bias does not pose a problem.

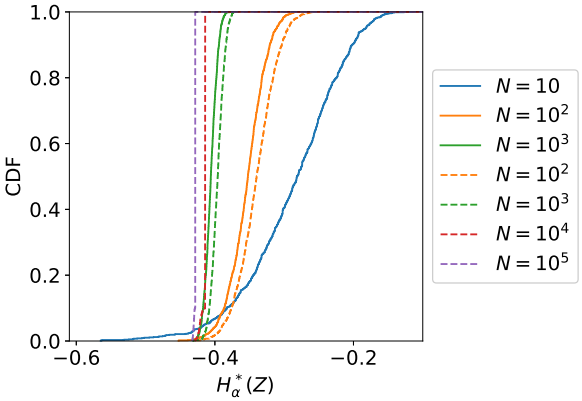
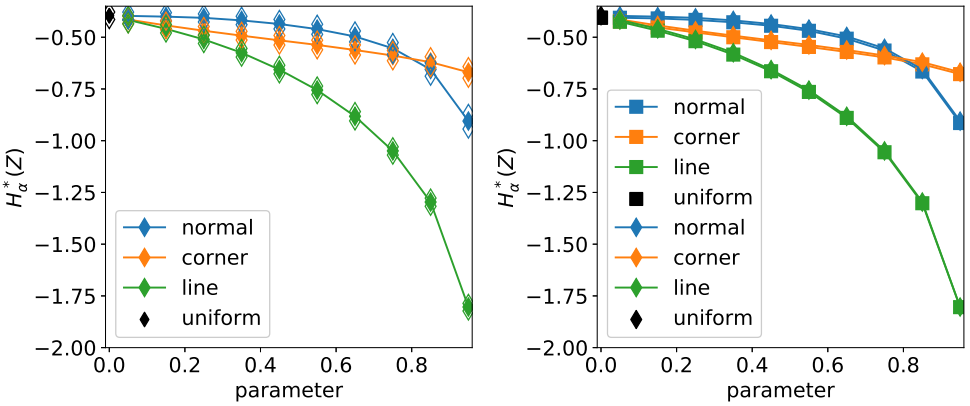


Figure 2.9: Empirical distribution for the uniform distribution for varying N. The solid lines refer to the distributions based on the MST, while the dashed lines refer to the distributions based on the FMST. Results using the MST method are limited to $N \leq 10^3$ because of high computational cost.



(a) Mean and confidence intervals for the FMST method.

(b) Comparison with the MST method.

Figure 2.10: Behavior of the FMST estimator for two different types of data distribution. On the left the mean and empirical 95%-confidence intervals, while on the right the means of the MST (squares) and FMST (diamonds) estimates can be compared (see also Figure 2.3).

2.4.1 Comparison with the exact value of the Rényi divergence

In this section, we present a simulation study to compare the true value of the dependence measure with the measures obtained with MST and FMST estimators. A direct method to estimate the Rényi mutual information based on a kernel density estimator (KDE) is given by [32] and this method is used for comparison.

The distribution under consideration is

$$X = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix}, \quad X \sim N(\mu, \Sigma), \quad \mu = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \Sigma = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix},$$

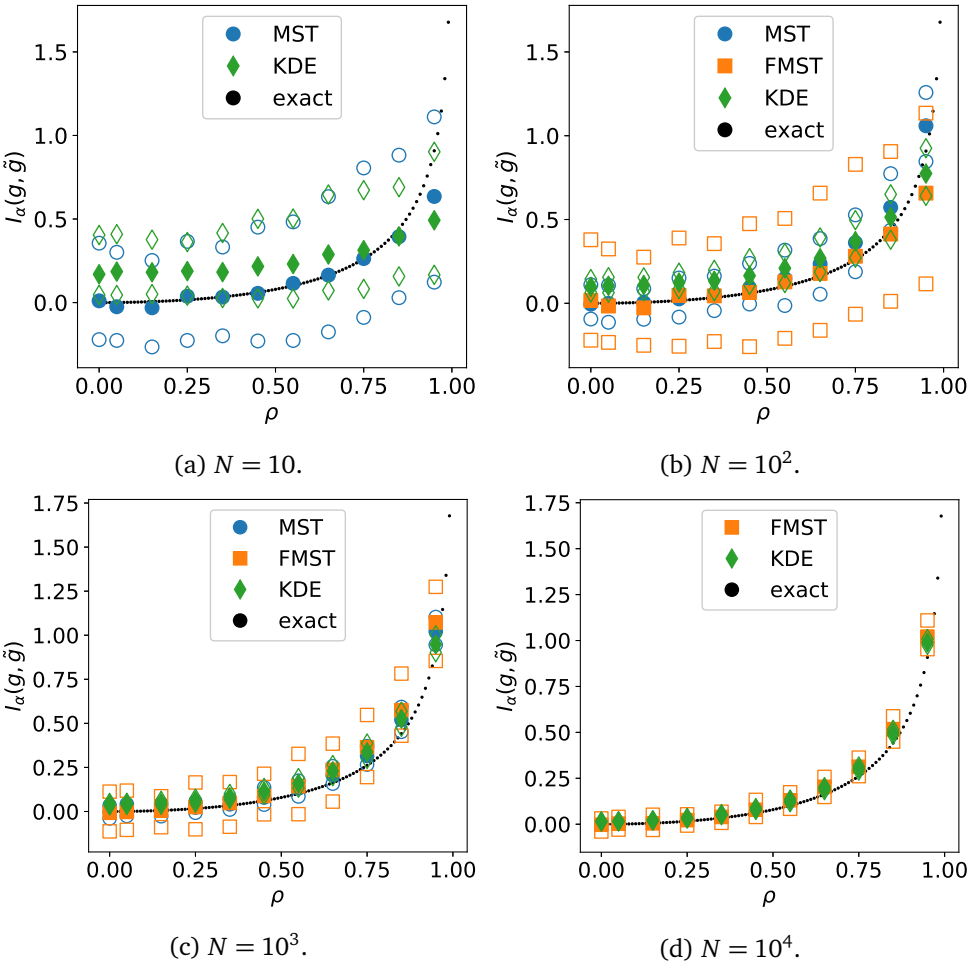
for which the Rényi mutual information is given by

$$I_\alpha = \frac{-1}{2(1-\alpha)} \left(\alpha \log(1-\rho^2) - \log(1-\alpha^2\rho^2) \right).$$

We vary the value of ρ and study the behavior of the MST and FMST estimator of the Rényi mutual information. A direct estimate is obtained via the estimator of [32] with a Gaussian kernel and a bandwidth computed with Scott's rule [45]. In this rule, we use $d = 1$ because the product kernel is the product of the two one-dimensional kernels. Because of symmetry, only positive values for ρ are considered.

Multiple sample sizes are included and the estimation is repeated n_r times, each time with newly generated data. We choose $n_r = 10^2$ for all methods. We give the average value in Figure 2.11 in solid markers, where it has to be noted that the means for the MST and FMST methods often overlap. The open markers represent the empirical 95% confidence interval. The constant $\beta_{L,\gamma}$ is computed for the same n_r. It can be seen that KDE overestimates the divergence for small ρ in general. For large ρ, this depends on N. For small N, the methods underestimate, while they overestimate for large N. Also, KDE has lower spread than the (F)MST method, but covers the analytic value less often. In Figure 2.12, the time aspects of the different methods are considered. The MST method is left out for $N = 10^4$ because of its high computational cost.

For larger values of N, the time complexity of the FMST method and KDE is as expected, with complexity $O(N^{1.5})$ and $O(N^2)$, respectively. The FMST method becomes cheapest for $N = 10^4$ while its results are very similar. Therefore, it is beneficial to use the FMST method in the case of large data sets.



(a) $N = 10$. (b) $N = 10^2$.

(c) $N = 10^3$. (d) $N = 10^4$.

Figure 2.11: Comparison of the estimated value to the exact value of the divergence.

2.5 Test cases

The proposed methods are applied to two test cases in this section. In the first case, we use the Ishigami function [46] and evaluate it using randomly sampled synthetic data as inputs. We consider data from two different input distributions, one without dependencies (uniform) and one with strong dependencies. On all combinations of variables (input and output), the dependence is quantified and, for the uniform data set, compared to the values of the Sobol (total) indices. In the second test case, we investigate weather data obtained from the Royal Netherlands Meteorological Institute (KNMI).

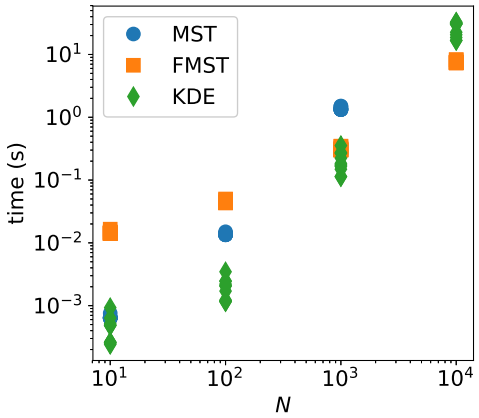


Figure 2.12: Time aspects for the different methods.

2.5.1 Ishigami function

This test case will run through the complete thesis to illustrate the different concepts. Its test function is from Ishigami & Homma [46] and its Sobol (total) indices [47] can be computed analytically. The test function is given by

$$I(x, y, z|a, b) = (1 + bz^4)\sin(x) + a\sin^2(y). \tag{2.10}$$

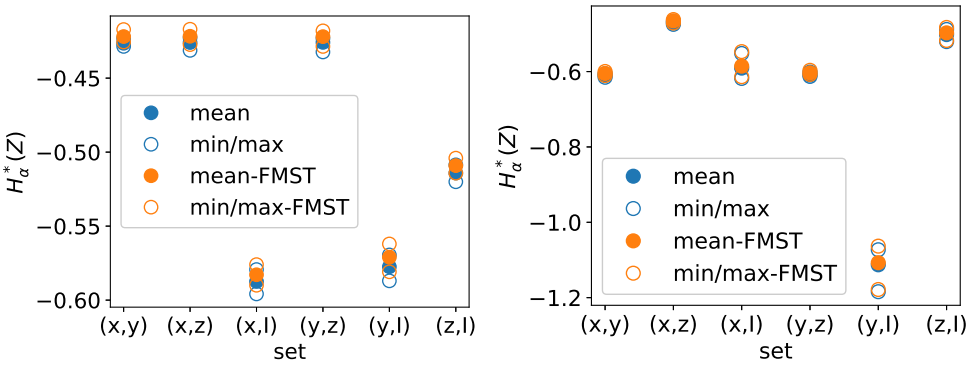
The parameters a and b are chosen to be 7 and 0.1, respectively, in accordance with [48]. One data set is three-dimensional and uniformly distributed on $[-\pi, \pi]^3$, while the other one is generated as

$$N\left(\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}1 & 0.8 & 0.5\\0.8 & 1 & 0.8\\0.5 & 0.8 & 1\end{bmatrix}\right), \tag{2.11}$$

and range-normalized to $[-\pi, \pi]^3$. The MSTs are approximated with both the MST and the FMST method and (2.9) is used to compute the dependencies both between input variables and between the input variables and the output variable.

The data sets have been generated $n_r = 10$ times with $N = 10^4$ samples each and the results are in Figure 2.13. First of all, it can be seen that the estimates are robust, as for each set (or pair) the estimates are all in a narrow interval (indicated by the minimum and maximum estimates).

Furthermore, it can be seen that for the uniform data set (Figure 2.13a), only combinations of input and output are dependent, while combinations of input variables are independent. In case of independence, the value is around -0.42. One can compare this with Figure 2.9, where it can be seen that in case of two



(a) Uniform data set.

(b) Dependent data set.

Figure 2.13: Estimates of $H_\alpha^*(Z)$ for two data sets. Note the difference in the range of the y-axis.

uniformly distributed independent variables, the estimate is slightly below -0.4 in case of $N = 10^4$. Indeed, the input variables are all independent in case of this uniform data set.

The dependence of I on the various inputs is illustrated in Figure 2.14 where scatter plots of the input-output pairs are shown. Note the difference between the plots of (x, I) and (y, I), while their estimates for the uniform case are similar.

For the strongly dependent data set (Figure 2.13b), the analysis is less straightforward. Here, the combination (y, I) shows the strongest dependency, while the estimate for (z, I) stays approximately the same. The relative ordering of input-output dependencies is consistent with the intuition gleaned from the right panels of Figure 2.14. Because of the structure of the data set, all input variables are mutually dependent. The dependencies (x, y) and (y, z) have a similar value as expected, while the combination (x, z) has a weaker dependency and is nearly independent.

We continue by comparing the means of the estimates for the input-output combinations for the uniform data set based on FMST with the values of its Sobol indices and Sobol total indices in Table 2.1. For the Ishigami function, the sum of

Table 2.1: Comparison of the proposed estimator and Sobol indices for the Ishigami function with independent (uniformly distributed) input variables.

	S_i	S_{Ti}	$H_\alpha^*(Z)$
x	0.314	0.558	−0.583
y	0.442	0.442	−0.571
z	0	0.244	−0.509

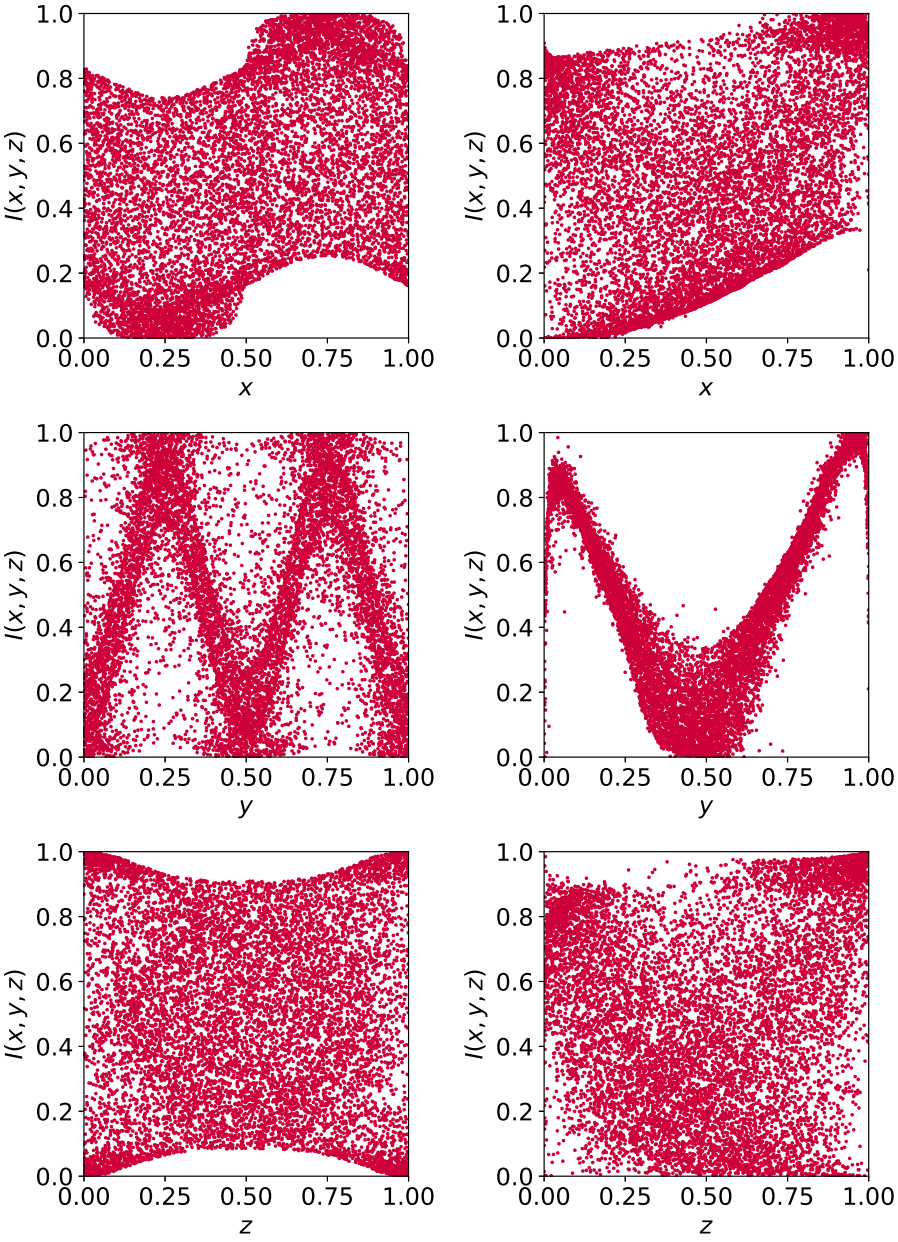


Figure 2.14: Scatterplots for the Ishigami function with on the left the uniform data set and on the right the dependent data set. The dependent input data has an effect on the output distribution.

all S_i and first-order interaction terms S_{ij} equals 1, and only S_x, S_y and S_{xz} are nonzero. Since $S_y = S_{Ty}$, y is not involved in interaction terms, while S_z does not have an effect by itself. The term S_{xz} contributes to both S_{Tx} and S_{Tz}. From the numerical values for the Sobol indices, summarized in Table 2.1, we conclude that y is most important by itself, while x is most important when interactions with other variables are included.

The proposed estimator $H^*_\alpha(Z)$ behaves different from S_i. The estimator $H^*_\alpha(Z)$ does not have the sum property, nor does it compute estimates for interaction effects. It is approximately -0.42 in case of independence (exact value depends on N, higher N leads to a smaller value, see Figure 2.9), and drops below this value if the variables are dependent. The values of $H^*_\alpha(Z)$ in Table 2.1 indicate x and y are nearly equally important. In the ordering, it is consistent with the total indices.

In practice, it might be beneficial to compute all the Sobol indices when the input variables are independent and not large in number. However, our prime interest in this study is in general cases with dependent input variables given to us in the form of data sets. In these cases, Sobol indices are difficult to calculate, especially if one wants to take the dependencies into account. By contrast, $H^*_\alpha(Z)$ can be computed easily for dependent variables. For dependent variables, the expressions for the Sobol indices become more comprehensive and the interpretation becomes less straightforward, due to the possibility of negative values for the indices.

2.5.2 KNMI data

In this test case, we investigate data obtained from the Royal Netherlands Meteorological Institute (KNMI). This data concerns weather observations at a location in the North Sea, to be more precise: the Lichteiland Goeree. Each hour, several variables such as wind speed and direction are recorded. We use the data from January 1st 1981 to December 31th 2010. The variables we used are in Table 2.2. In this data, we find several complications which can occur in real-life situations. We will name them shortly. First, the presence of discrete variables due to measurement accuracy. This can be handled in the rank-transform, but requires some attention from the analyst. In this section, we have placed the data points with the same values at the same location on the axis, which is determined by computing the average of the location of these data points as would be the case if they were nonidentical. After the rank-transform, one can add a small noise to separate the data points, as data points at the same location may lead to problems in the minimum spanning tree algorithm, depending on the implementation used. The same holds for actual discrete values (second complication). Another complication is when indicator values are present. This occurs for example in MWD, where there

Table 2.2: Variables included in the analysis.

Abbreviation	Description
MWD	Mean wind direction (in degrees) during the 10-minute period preceding the time of observation
MWS60	Hourly mean wind speed (in 0.1 m/s)
MWS10	Mean wind speed (in 0.1 m/s) during the 10-minute period preceding the time of observation
MWG	Maximum wind gust (in 0.1 m/s) during the hourly division
T	Temperature (in 0.1 degrees Celsius) at 1.50 m at the time of observation
TDP	Dew point temperature (in 0.1 degrees Celsius) at 1.50 m at the time of observation
AP	Air pressure (in 0.1 hPa) reduced to mean sea level, at the time of observation
HV	Horizontal visibility at the time of observation
RAH	Relative atmospheric humidity (in percents) at 1.50 m at the time of observation
SST	Sea surface temperature (in 0.1 degrees Celsius) at the time of observation

are separate values for no wind or variable wind direction. We decide to leave those measurement points out of the analysis. Incomplete observations are also removed from the data set. This left us with $N = 142114$ data points in 10 input variables.

We quantify the dependency strengths as $D_\alpha(g, \tilde{g})$ (2.3) by (2.5) and (2.6) with the FMST method. To estimate the Rényi entropy by (2.6) for all combinations of variables, we also need to obtain an estimate for $\beta_{L,\gamma}$, which we get from $n_r = 100$ samples of uniform bivariate data with the same number of samples. The values are in Figure 2.15. All combinations of two variables show dependencies, where temperatures (T, TDP, SST) and air pressure (AP) are least related. Strongest dependencies exist between the wind speeds (MWS60, MWS10, MWG) and between them and the visibility (HV). The dependencies between the wind speeds are expected, while the connection between them and the visibility is caused by fog formation being associated with weak wind speeds [49].

2.6 Discussion

In this study, we have combined several methods to come to an efficient approach for quantifying dependencies in data by means of Rényi mutual information. With

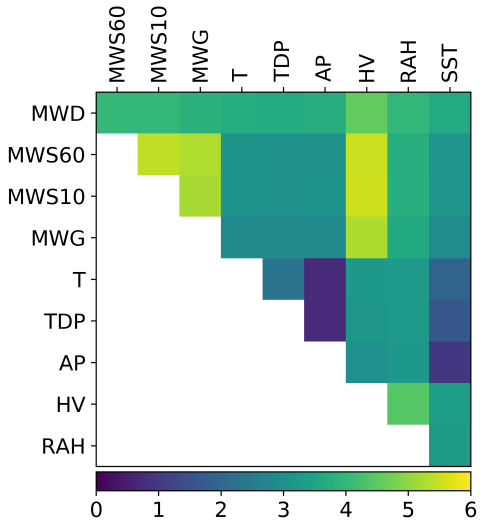


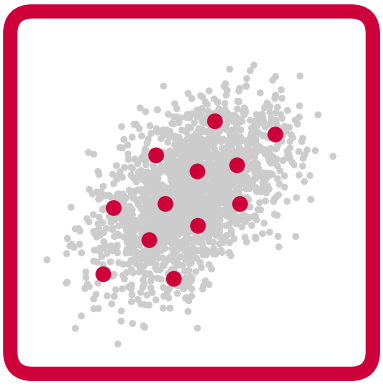
Figure 2.15: Dependency analysis for weather data. The values for $D_\alpha(g, \tilde{g})$ are shown in the heatmap.

this approach, large multivariate data sets can be handled. The Rényi mutual information is used for the quantification, in which the mutual information is estimated via a minimum spanning tree (MST). The MST can be approximated in order to speed up the computation. A major advantage of this combination is that it does not require estimation (or a priori knowledge) of probability densities.

Furthermore, we have proposed a novel estimator for the case where the ranking of the dependencies is required rather than their exact strength. In this case, the computation of a constant bias term can be omitted.

We have tested our approach on the Ishigami function as well as on a real-world test case involving weather data. In both cases, the results partially reflected prior knowledge or heuristic understanding, but also showed unexpected results which could be explained by the data. Moreover, in the Ishigami test case we included an example with independent input variables, making it possible to compare our approach with the analysis based on Sobol indices. The results from our proposed method are consistent with those from the Sobol total indices. However, the Sobol indices are not suitable at all for dependent data.

Altogether, the approach proposed here is a suitable and useful method to quantify dependencies. It gives consistent results and remains computationally feasible even for large data sets by using the FMST algorithm. This method will be extended to a sensitivity analysis method by constructing sensitivity indices from the Rényi divergence in Chapter 5. An interesting aspect of this extension will be to separate in these indices direct effects from indirect effects.



3 Sample selection

In the previous chapter, we have shown how to measure dependencies in the input data. We now continue to the next step, namely the sample selection. It is often assumed that the input variables are independent with known probability distribution function, which is not the case in our setting. Therefore, we propose in this chapter new sample selection methods which do not require independence or known input distributions.

3.1 Introduction

Uncertainty propagation is a core topic of uncertainty quantification (UQ). This topic involves the propagation of uncertainties from the input variables to the output variables. When the computational model is considered to be a black-box, the key challenge is sample selection, i.e., how to select the input samples for which the computational model is evaluated. Related questions are encountered in many fields of science and engineering [11, 50–53] and have given rise to modern UQ methods including stochastic collocation, polynomial chaos expansion and stochastic Galerkin methods [6, 54–57].

A still outstanding challenge is how to characterize model output distributions efficiently in case of multivariate, dependent input distributions. Independence between the inputs is assumed in the previously mentioned methods, e.g., for the construction of the Lagrange polynomials in stochastic collocation, or for the construction of the orthogonal polynomials in generalized polynomial chaos. When independence between the input variables holds, the multivariate problem can be factored easily into multiple one-dimensional problems. The one-dimensional solutions can be combined to a solution for the multi-dimensional problem by

Mainly based on A. Eggels, D. Crommelin, and J. Witteveen, "Clustering-based collocation for uncertainty propagation with multivariate dependent inputs," *International Journal for Uncertainty Quantification*, vol. 8, no. 1, pp. 43–59, 2018.

tensor products. When the inputs are dependent, such factorization can become extremely complicated if the inputs have non-Gaussian distributions, making it unfeasible in practice for many cases. It generally involves nontrivial transformations that require detailed knowledge of the joint distribution (e.g. Rosenblatt transformation [34]). Such information is often not available. In [58], factorization is circumvented and instead the problem is tackled by using the Gram-Schmidt orthogonalization procedure to get an orthogonal basis of polynomials, in which the orthogonality is with respect to the distribution of the inputs. However, this procedure gives non-unique results that depend on the implementation.

In this chapter we propose a novel approach for efficient UQ with multivariate, dependent inputs. This approach is related to stochastic collocation. It employs collocation nodes obtained from data clustering though, rather than from constructing a standard (e.g. Gaussian) quadrature or cubature rule. By using techniques from data clustering, we can construct sets of nodes that give a good representation of the input data distribution, well capable of capturing correlations and nonlinear structures in the input distributions. It is straightforward to obtain weights associated with these nodes. All weights are guaranteed to be positive.

The approach we propose is non-intrusive and able to handle non-Gaussian dependent inputs. We demonstrate that it remains efficient for higher dimensions of the input, notably in case of strong dependencies. These dependencies are not limited to correlations (linear dependencies), but can also be nonlinear. Furthermore, the approach employs data clustering of an input data set. The underlying input distribution can be unknown, and there is no fitting of the distribution involved. Hence, no fitting error is introduced. This makes the approach particularly suitable for situations where the exact input distribution is unknown and only a sample of it is available.

We emphasize that the method we propose here does not employ orthogonal polynomials and their roots, nor does it require the specification of an input distribution. This constitutes a main difference with stochastic collocation. Furthermore, we demonstrate that generating a random quadrature rule, by randomly selecting points from the sample of inputs and using these as cluster centers, gives unsatisfactory results. This is due to the fact that such a random selection is ill-suited to sample or represent the tails of the input distribution.

The outline of this chapter is the following: in Section 3.2, we start by briefly summarizing stochastic collocation and multivariate inputs. We discuss the challenges of dealing with dependent inputs, and we introduce the concept of clustering-based collocation. In Section 3.3, we describe three different clustering techniques and give a convergence result for one dimension. In Section 3.4, we present results of numerical experiments in which we test our clustering-based collocation

method, using the clustering techniques described in Section 3.3. The conclusion follows in Section 3.5.

3.2 Stochastic collocation and its extension ⋆

Consider a function $u(\mathbf{x}) : \Omega \mapsto \mathbb{R}$, $\Omega \subseteq \mathbb{R}^d$, that maps a vector of input variables to a scalar output. Let us assume $\mathbf{x}$ is a realization of a random variable χ with probability density function $p(\mathbf{x})$. We would like to characterize the probability distribution of $u(\mathbf{x})$, in particular we would like to compute moments of $u(\mathbf{x})$:

$$\mathbb{E}[u^q] = \int_\Omega (u(\mathbf{x}))^q p(\mathbf{x}) \mathrm{d}\mathbf{x}.$$

In what follows, we focus on the first moment:

$$\mu := \mathbb{E}[u] = \int_\Omega u(\mathbf{x}) p(\mathbf{x}) \mathrm{d}\mathbf{x}. \tag{3.1}$$

We note that higher moments can be treated in the same way, as these are effectively averages of different output functions, i.e. $\mathbb{E}[u^q] = \mathbb{E}[v]$ with $v(\mathbf{x}) := (u(\mathbf{x}))^q$. In both cases, the expectation is with respect to the distribution of χ. In stochastic collocation, the integral in (3.1) is approximated using a quadrature or cubature rule. As is well-known, a high degree of exactness of the integration can be achieved for polynomial integrands with Gaussian quadrature rules.

3.2.1 Multivariate inputs

For multivariate inputs ($d > 1$), stochastic collocation based on Gaussian quadrature can be constructed using tensor products if the input variables are mutually independent. In this case, we can write $f(\mathbf{x})$ as a product of one-dimensional probability density functions. The degree of exactness of the corresponding cubature rule is $2m-1$ in each dimension if m collocation nodes are used in each input dimension. This requires a number of nodes (m^d) that grows exponentially in d. So, collocation with full tensor grids suffers from the curse of dimensionality. To reduce the number of nodes, Smolyak sparse grids [59, 60] can be used. The construction of Smolyak sparse grids will not be explained in detail here, but an important aspect is that the resulting set of nodes is a union of subsets of full tensor grids. When the grids are nested and the number of nodes in the nth level for one dimension, m_n^1, is $O(2^n)$, then the number of nodes in d dimensions scales as $O(2^n n^{d-1})$ [60], in contrast to $O(2^{nd})$ for the corresponding tensor rule.

3.2.2 Gaussian cubature with dependent inputs

As already mentioned, tensor grids are useful for stochastic collocation in case of independent inputs. If the input variables are dependent, grids constructed as tensor products of one-dimensional Gaussian quadrature nodes no longer give rise to a Gaussian cubature rule. In [58], generalization to dependent inputs is approached by constructing sets of polynomials that are orthogonal with respect to general multivariate input distributions, using Gram-Schmidt orthogonalization. The roots of such a set of polynomials can serve as nodes for a Gaussian cubature rule.

With the approach proposed in [58], the advantages of Gaussian quadrature (in particular, its high degree of exactness) carry over to the multivariate, dependent case. However, one encounters several difficulties with this approach. First of all, for a given input distribution, the set of obtained nodes is not unique. Rather, the resulting set depends on the precise ordering of the monomials that enter the Gram-Schmidt procedure. For example, with two-dimensional inputs and cubic monomials, 24 different sets of nodes can be constructed, as demonstrated in [58]. It is not obvious a priori which of these sets is optimal.

A further challenge is the computation of the weights for the cubature rule. In one dimension, they are computed by constructing Lagrange interpolating functions and evaluating their integrals. It is not straightforward how to do this in multiple dimensions. The alternative for computing the weights is to solve the moment equations. However, the resulting weights can be negative. Furthermore, one cannot choose the number of nodes freely: in general, with input dimension d and polynomials of degree p, one obtains p^d nodes. Thus, the number of nodes increases in large steps, for example with $d = 8$ the number of nodes jumps from 1 to 256 to 6561, respectively, if p increases from 1 to 2 to 3. It is unknown how to construct useful (sparse) subsets of nodes from these.

3.2.3 Clustering-based collocation

To circumvent the difficulties of Gaussian cubature in case of dependent inputs, as summarized in the previous section, we propose an alternative approach to choose collocation nodes. By no longer requiring the collocation nodes to be the nodes of an appropriate Gaussian cubature rule, we do not benefit anymore from the maximal degree of exact integration associated with Gaussian quadrature or cubature. However, we argue below that this benefit of Gaussian cubature offers only limited advantage in practice.

If one has a sample of the inputs available but the underlying input distribution is unknown, the Gaussian cubature rule will be affected by the sampling error (via the Gram-Schmidt orthogonalization). Alternatively, if the input distribution

is estimated from input sample data, the precision of the Gaussian cubature rule is also limited by the finite sample size.

Additionally, the degree of exactness is strongly limited by the number of nodes in higher dimensions. For example, suppose one can afford no more than 256 evaluations of the output function $u(\mathbf{x})$ because of high computational cost, i.e., one can afford a Gaussian cubature rule with 256 nodes. This gives very high degree of integration exactness (degree 511) in one dimension ($d = 1$), but the degree of exactness decreases to 31, 7 and 3, respectively, as the input dimension d increases to 2, 4 and 8. For $d > 8$, the degree of exactness is only 1 in case of 256 nodes, so only linear functions can be integrated exactly. The number of nodes for a full tensor grid in d dimensions with 2^n nodes in level n for one dimension is 2^{nd}, while a corresponding Smolyak grid contains $O(2^n n^{d-1})$ points. However, the approximation accuracy for the full grid is $O(2^{-ns})$ and $O(2^{-ns} n^{(d-1)(s+1)})$ for the sparse grid with $O(2^n)$ nodes in level n for one dimension [60], where s is the smoothness of the function (being the number of derivatives it has). This is still limiting for a high number of dimensions.

Furthermore, the accuracy of the propagation method does not need to be higher than the accuracy of the input uncertainty. Since the input is given by samples, the high accuracy of spectral methods is not fully utilized.

Instead of constructing a Gaussian cubature rule, we aim to determine a set of nodes that are representative for the sample of input data or for the input distribution, with the locations of the nodes adjusting to the shape of the distribution. Clustering is a suitable method (or rather collection of methods) to achieve this objective and will be explained in more detail in Section 3.3.

The basic idea, in the context of this study, is the following. Assume we have a data set $\{\mathbf{x}_1, ..., \mathbf{x}_N\}$ available, with $\mathbf{x}_i \in \mathbb{R}^d$. We define a partitioning of $\mathbb{R}^d$ existing of L subsets, denoted Ω_l with $l = 1, ..., L$. A cluster is a subset of the data falling into the same Ω_l. A common way to define cluster centers z_l is as the average of the data in each cluster, i.e.

$$\mathbf{z}_l := \frac{\sum_{i=1}^{N} \mathbf{x}_i \, \mathbf{1}(\mathbf{x}_i \in \Omega_l)}{\sum_{i=1}^{N} \mathbf{1}(\mathbf{x}_i \in \Omega_l)}, \tag{3.2}$$

with $\mathbf{1}(\cdot)$ the indicator function. If we define weights w_l as the fraction of all the data falling in the j-th cluster, that is,

$$w_l := \frac{\sum_{i=1}^{N} \mathbf{1}(\mathbf{x}_i \in \Omega_l)}{N}, \tag{3.3}$$

the weighted average $\bar{\mathbf{z}} := \sum_l w_l \, \mathbf{z}_l$ equals the data average $\bar{\mathbf{x}} := N^{-1} \sum_i \mathbf{x}_i$. Thus, $\bar{\mathbf{z}} = \bar{\mathbf{x}}$ by construction.

The key idea of what we propose here is to carry out collocation based on clustering of the input data. More specifically, we propose to use the cluster centers $\mathbf{z}_l$ and weights w_l as the nodes and weights of a quadrature rule. Thus, the (exact) first moment of the output function $u(\mathbf{x})$ over the input data is

$$\mu = \frac{1}{N}\sum_{i=1}^{N} u(\mathbf{x}_i), \tag{3.4}$$

and the approximation using clustering-based collocation is

$$\hat{\mu} := \sum_{l=1}^{L} w_l\, u(\mathbf{z}_j). \tag{3.5}$$

We emphasize that the number of function evaluations in (3.4) and (3.5) is different. When $L \ll N$, large savings in computational time can be achieved due to the smaller amount of evaluations of $u(\mathbf{x})$.

The proposed approximation (3.5) to estimate the first moment of $u(\mathbf{x})$ does not explicitly consider a function approximation of $u(\mathbf{x})$. However, (3.5) can be seen as the Monte Carlo integral over a function approximation of $u(\mathbf{x})$ which is piece-wise constant on the clusters.

It is easy to show that the approximation is exact ($\hat{\mu} = \mu$) for all linear input functions, due to the fact that $\bar{\mathbf{z}} = \bar{\mathbf{x}}$, as mentioned above. In other words, the degree of exactness is one: we can consider (3.5) as a quadrature rule for the integral of $u(\mathbf{x})$ over the empirical measure induced by the data set $\{\mathbf{x}_1, ..., \mathbf{x}_N\}$. This quadrature rule is exact if $u(\mathbf{x})$ is linear. This may seem limited in comparison to Gaussian quadrature, but as discussed earlier, the degree of exactness of Gaussian quadrature reduces rapidly if the input dimension d grows and the number of nodes remains constant. For nonlinear input functions, the approximation (3.5) will in general not be exact. We will investigate its convergence in Sections 3.3.5 and 3.4.

3.3 Clustering methods

In this section, we describe three different methods to construct clusters, i.e. three methods to construct a suitable collection of subsets Ω_j. As already mentioned, the methods are based on input given as a data set in d dimensions with N data points $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ with $\mathbf{x}_i \in \mathbb{R}^{1\times d}$ also denoted by a matrix $X \in \mathbb{R}^{N\times d}$. If the input is given as a distribution, we can create a data set by sampling from this distribution. Furthermore, we scale this data set to $[0,1]^d$ by linear scaling with the range. This is done to comply with the domain of the test functions we will use further on.

We cluster the data points into L clusters $\{C_1,\dots,C_L\}$ with centers $\{\mathbf{z}_1,\dots,\mathbf{z}_L\}$ and use these as nodes. The centers are computed as the means of the data points in the corresponding cluster, see (3.2). We investigate three different methods, namely k-means clustering, principal component analysis based clustering and a method with randomly selected data points as cluster centers. In the following, we will use the words clustering and partitioning interchangeably.

3.3.1 k-means clustering

The k-means method is one of the oldest and most widely used methods for clustering [44,61]. The idea behind it is to minimize the within-cluster sum-of-squares (SOS):

$$\min_{\{\mathbf{z}_1,\dots,\mathbf{z}_L\}} SOS(\mathbf{z}_1,\dots,\mathbf{z}_L), \quad SOS(\mathbf{z}_1,\dots,\mathbf{z}_L) = \sum_{i=1}^{N} ||\mathbf{x}_i - \mathbf{z}_{\text{argmin}_l ||\mathbf{x}_i-\mathbf{z}_l||_2^2}||_2^2,$$

in which $\mathbf{z}_l$ are the cluster centers and $\mathbf{x}_i$ the data points. The minimization problem is solved with an iterative procedure. The cluster centers $\mathbf{z}_l$ are initialized randomly, after which, in each step of the algorithm, all data points are assigned to the nearest cluster center (in the Euclidean norm) and the cluster centers are recomputed as the mean of their assigned data points. The algorithm finishes when the cluster centers do not move anymore. There are many extensions and improvements of the (initialization of the) algorithm, such as using the triangle inequality to avoid unnecessary distance calculations [62], the use of global methods [63–65] and low-rank approximations [66]. We will use the k-means++ initialization method, as described in [67].

Because the algorithm contains a random initialization and the objective function is nonconvex, it can converge to a local minimum rather than to the global optimum. Therefore, in our numerical tests in Section 3.4, the algorithm is performed r times ($r > 1$) with different initializations and the best solution (with minimal SOS) is chosen. We use a fixed number of iterations in the minimization. In some cases, the iterations have not fully converged yet. This will be ignored because, in practice, nearly all of the r executions converge so that the chosen best solution is always a converged solution. Further onwards, we will refer to this method as KME. We choose $r = 10$.

3.3.2 PCA-based clustering

With this method, based on [68,69] and principal component analysis (PCA), one starts with a single large cluster containing all the data, and in each step, the cluster with the largest average radius is split into two. This is implemented by

splitting the cluster whose data points have the largest average squared Euclidean distance to the cluster center. We split such that the cutting plane goes through the old cluster center (center of mass) and is perpendicular to the largest principal component of the covariance matrix of the data in the cluster, as suggested by [69]. This continues until the desired number of clusters L is attained. We note that other stopping criteria can be used as well, but these are less useful for the purpose of this study.

This method (referred to as PCA-based later on) is deterministic, unlike the k-means method described in the previous section. Clustering by the diameter criterion is already performed in [70], but there the cluster with the largest diameter is split. Division methods based on farthest centroids are suggested in [71]. Other refinements of this method are also possible, e.g., the merging of clusters at some steps in the algorithm, but we will not explore these here. These can be investigated in future work.

A possible disadvantage of this method is that there is no guarantee that all data points end up in the cluster with the nearest cluster center. This because the splits are performed independently and data points from other clusters are not reassigned if a closer cluster center appears. There are two ways to fix this. The first, cheap option is to reassign all data points to the nearest cluster center at the end of the method. The second, expensive option is to do this in each step of the method. We will implement the first option (reassigned principal component analysis, denoted PCR) as well as the original option of not reassigning.

3.3.3 Random clustering

For comparison purposes we include a third method, in which cluster centers are selected randomly. This method consists of randomly selecting data points from the data set, all with equal probability, and use these as cluster centers. The clusters are formed by assigning each data point to its nearest cluster center. This method will be referred to as MCC (Monte Carlo clustering).

3.3.4 Calculation of weights

As already mentioned, we use the cluster centers as nodes for collocation. To do so, each node must be assigned a weight. In all three methods, the weights are determined by the number of data points in the cluster associated with that node, divided by the total number of data points, see (3.3). By construction, all weights are positive and their sum equals one.

3.3.5 Convergence ⋆

In the case of one dimension ($d = 1$), it can be proven that the PCA-based clustering converges to the Monte Carlo integral for increasing values of L. The proof relies on the fact that in each step, the largest cluster radius either decreases or remains constant. If $d = 1$, the largest cluster radius equals

$$\delta^*(L) = \max_{i\in\{1,\dots,N\}} \min_{l\in\{1,\dots,L\}} \{|x_i - z_l|\}.$$

As can be seen, it depends on L. We give the proof for one dimension.

We can define a finite interval $D := [x_-, x_+]$ which contains all the data $\{x_1, \dots, x_N\}$. Furthermore, we assume that the output function $u(x)$ is Lipschitz continuous on this interval with Lipschitz constant L^*. Let ν be the empirical measure on D, i. e. $\nu(\Omega \subseteq D) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}(x_i \in \Omega)$ for each subset Ω of D. Suppose that we have for each $L \in \mathbb{N}^+$, $L \leq N$ that $x_- < z_1 < z_2 < \dots < z_L < x_+$ are the ordered cluster centers. A set partitioning is defined by $\cup_{l=1}^{L} E_l$ with $E_l := \left[\frac{z_{l-1}+z_l}{2}, \frac{z_l+z_{l+1}}{2}\right)$, for $l = 2, \dots, L-1$, $E_1 = \left[x_-, \frac{z_1+z_2}{2}\right)$ and $E_L = \left[\frac{z_{L-1}+z_L}{2}, x_+\right]$. Define $u_l(x) := u(z_l)$, $\forall x \in C_l$ and 0 elsewhere for $l = 1, \dots, L$. Now, denote $\tilde{u}(x) = \sum_{l=1}^{L} u_l(x)$. Because of the Lipschitz continuity, we have that

$$\begin{aligned} &\forall\, x_i \in D\ \exists\, l = l(x_i) \in \{1, \dots, L\} : \\ &\qquad |u(x_i) - \tilde{u}(x_i)| = |u(x_i) - u(z_l)| < L^*\delta^*(L). \end{aligned}$$

In the PCA-based algorithm for $d = 1$, δ^* is strictly non-increasing as L grows. It reaches its lower bound $\delta^*(L) = 0$ when $L = N$, because then each data point is its own cluster center. We can now bound the difference between the PCA-based integral $I_{\mathrm{PCA}}(L) = \sum_{l=1}^{L} u(z_l) w_l$ and the Monte Carlo integral $I_{\mathrm{MC}}(N) = \sum_{i=1}^{N} u(x_i)\frac{1}{N}$ as follows

$$\begin{aligned} |I_{\mathrm{PCA}} - I_{\mathrm{MC}}| &= \left| \frac{1}{N}\sum_{i=1}^{N} u(x_i) - \sum_{l=1}^{L} u(z_l) w_l \right|, \\ &= \left| \frac{1}{N}\sum_{i=1}^{N} u(x_i) - \frac{1}{N}\sum_{i=1}^{N}\sum_{l=1}^{L} u(z_l)\mathbf{1}(x_i \in \Omega_l) \right|, \\ &= \frac{1}{N}\left| \sum_{i=1}^{N} \left(u(x_i) - \sum_{l=1}^{L} u(z_l)\mathbf{1}(x_i \in \Omega_l) \right) \right|, \\ &\leq \frac{1}{N}\sum_{i=1}^{N} \left| u(x_i) - \sum_{l=1}^{L} u(z_l)\mathbf{1}(x_i \in \Omega_l) \right|, \\ &< \frac{1}{N}\sum_{i=1}^{N} L^*\delta^*(L) = L^*\delta^*(L). \end{aligned}$$

Since $\delta^*(l+1) \leq \delta^*(l)$ for all $l \in \mathbb{N}$ when $d = 1$, $\delta^*(l) \geq 0$ and $\delta^*(l = N) = 0$, the bound becomes stricter for increasing l.

For higher dimensions, the derivation of the bounds is analogous, although δ^* will not be monotonically decreasing, but it will decrease in general. This is also the case for the other methods, even for $d = 1$, where the nodes are not nested such as in the PCA-based case, such that it is not guaranteed that δ^* decreases monotonically. For the higher-dimensional case, we refer to Section 3.4.3 where we give a numerical result on convergence.

3.4 Results

We test the quadrature based on the clustering methods described in Section 3.3 by integrating the Genz test functions on the domain $[0,1]^d$ for three different data sets. In two of these data sets, the variables are mutually dependent. The relative error

$$\varepsilon = \frac{|I_{\text{CP}} - I_{\text{MC}}|}{|I_{\text{MC}}|}, \tag{3.6}$$

defined by the absolute difference between the integral calculated by the cluster points and weights (I_{CP}) and the Monte Carlo integral of the data, divided by the value of the Monte Carlo integral, is used as the measure of accuracy. We perform the MCC method $n_r = 10$ times for each setting to investigate the effect of randomness. We show the mean, minimum and maximum error for MCC. For comparison, we add results from using Monte Carlo sampling (MCS), which is repeated 10 times as well.

The first test is to assess how these methods perform under an increasing number of dimensions and what the effect of dependent variables is. Then, we compare the numerical convergence of the PCA-based method with the MCC and MCS methods for an increasing number of clusters. Finally, we compare the computational cost of these methods.

Furthermore, in Section 3.4.4 we test the proposed methods on the Ishigami function with two input data sets.

3.4.1 Genz test functions

Genz [72] has developed several functions to test the accuracy of a cubature rule. The definitions, our choice of parameters and some illustrations are given in Appendix A. We test the methods by integrating the Genz test functions over three different data sets consisting of $N = 10^5$ samples. We compare the results from Monte Carlo integration with the results obtained by clustering-based quadrature by the error measure (3.6).

3.4.2 Data sets

We use three data sets with different types of nonlinear relationships to illustrate the methods. All sets consist of $N = 10^5$ samples drawn from a certain distribution. The dimension d is allowed to vary from 1 to 16. The first set is the independent uniform distribution in d dimensions, the second set is a multivariate Gaussian distribution in d dimensions, and the third set is an artificial data set which contains strongly nonlinear relationships between the variables. All the data is range-transformed after generation to fit the domain $[0,1]^d$ of the Genz test functions. Their parameters are given as follows.

The uniform distribution has density 1 over the unit hypercube. The multivariate Gaussian distribution has means 0, variances 1 and correlation coefficients σ_{ij} between dimensions i and j given by

$$\sigma_{ij} = \frac{1}{2} \text{ for } |i-j| = 1, \quad \sigma_{ij} = 0 \text{ for } |i-j| > 1.$$

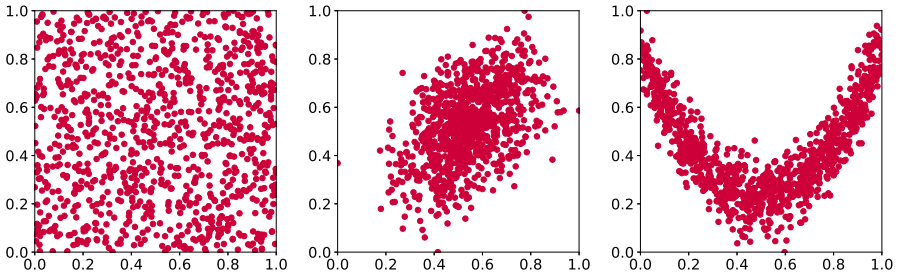
The third and last distribution is given as

$$\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_d \end{bmatrix} = \begin{bmatrix} U(-2,2) \\ X_1^2 \\ \vdots \\ X_1^d \end{bmatrix} + \sigma \begin{bmatrix} 0 \\ N(0,1) \\ \vdots \\ N(0,1) \end{bmatrix},$$

in which $U(-2,2)$ is the uniform distribution on $[-2,2]$, σ equals 0.5 and $N(0,1)$ is the standard normal distribution. We refer to this distribution as the "polynomial distribution".

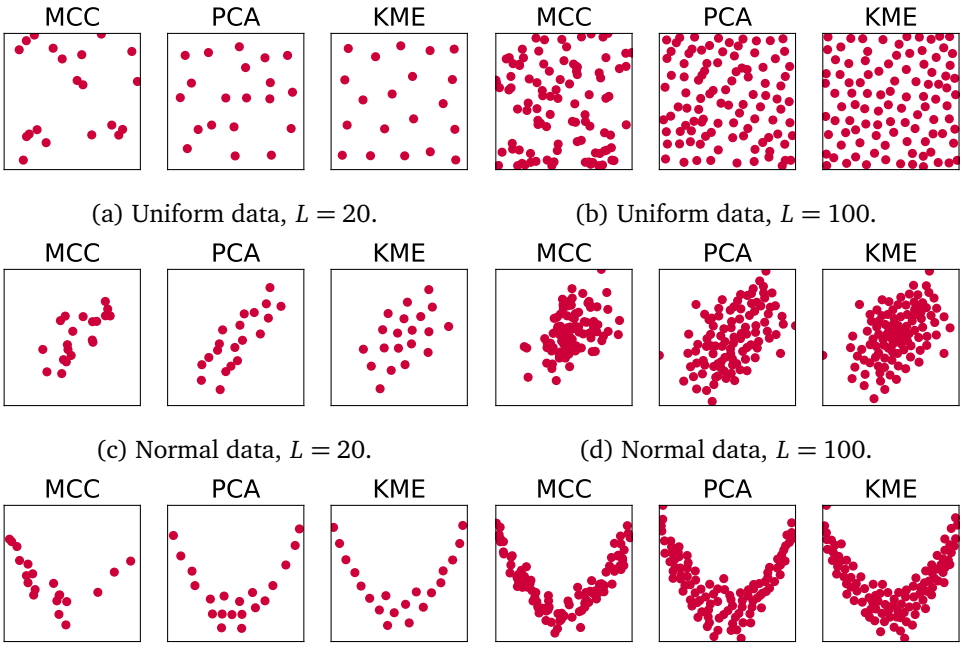
In Figure 3.1, we show 10^3 data points generated for $d = 2$ for the different test sets. From the figure, it is clear that these data sets have different types of nonlinear relationships. The uniform distributed data is independent, the normally distributed data is weakly dependent and the polynomial data contains strong nonlinear relationships between the variables and is far from Gaussian.

In Figure 3.2, the partitionings (for $L = 20$ and 100) for the different test sets are shown. One of the observations is that the MCC method yields most clusters in dense regions, just as the KME method. In the latter, the spacing between the nodes is more evenly distributed in space. However, the PCA-based method also has nodes in less dense regions of the data set and is even more evenly distributed.



(a) Uniform data. (b) Normal data. (c) Polynomial data.

Figure 3.1: Visualization of the test sets for $d = 2$ and $N = 10^3$. The uniform distributed data is independent, while the normal distributed data is weakly dependent and the polynomial data is strongly, nonlinearly dependent.



(a) Uniform data, $L = 20$. (b) Uniform data, $L = 100$.

(c) Normal data, $L = 20$. (d) Normal data, $L = 100$.

(e) Polynomial data, $L = 20$. (f) Polynomial data, $L = 100$.

Figure 3.2: Visualization of the partitionings for $d = 2$. The general observation is that MCC and KME have most nodes in dense regions of the data, while PCA-based is more spread out over the domain of the data.

3.4.3 Tests

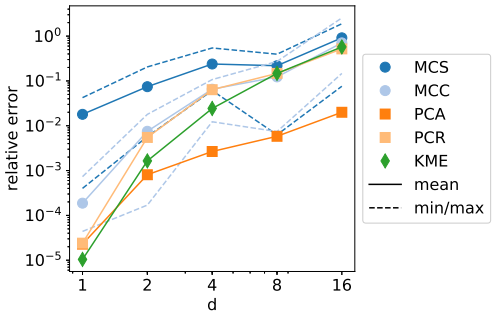
The tests of the methods will consist of integrating the test functions on each of the data sets and comparing the integrals to the Monte Carlo integrals. The data sets and cluster centers will be generated only once and reused for the different test functions. The output of each of the methods is the value of the integral of the test function when performed with the cluster points and weights. Not all results will be shown, but we will show representative examples. The error measure we use is the relative error defined by (3.6). The Monte Carlo methods are repeated 10 times due to their inherent randomness. The k-means method is random as well due to the chosen initialization, but here the repetition is included in the method itself.

Dimension effects

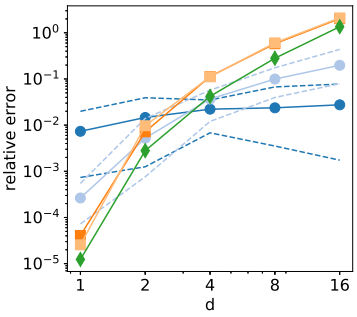
First, we compute the relative error as given by (3.6). We do this for various values of d and data sets, for a fixed number of cluster points $L = 50$, to see how the error relates to dimension. The results are in Figure 3.3 for the first and second Genz test function. For test function 1, the PCA and KME method are more accurate than the Monte Carlo methods. For test function 2, we see that for large d and independent or slightly dependent data, Monte Carlo may be the better choice because it does not suffer from the curse of dimensionality. A possible disadvantage here is the bandwidth of the Monte Carlo results. For both test functions, we see that reassigning data points to the nearest cluster center (PCR) does in general not improve the result of PCA. For the test functions 3-5, the results are similar (not shown). For the discontinuous test function 6, results are less robust (not shown), due to the discontinuity of the test function.

Effect of number of clusters

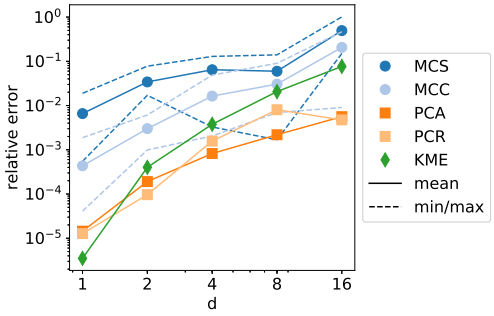
In Figure 3.4, the effect of increasing L is studied for $d = 4$. These results support the statements from Section 3.3.5, namely that the errors generally decrease with increasing L. We show the results for the first and second test function. Several observations from the previous section still hold: the bandwidth for Monte Carlo methods is large and the reassigning of the data points does in general not improve the result. When these methods are ignored, we see the PCA-based method gives the best results for test function 1, while for test function 2, this depends on the data set used. Note the contours of test function 1 are straight lines and this function is relatively easy. The rate of convergence is on average not higher than for MCS, which is $O(N^{-1/2})$.



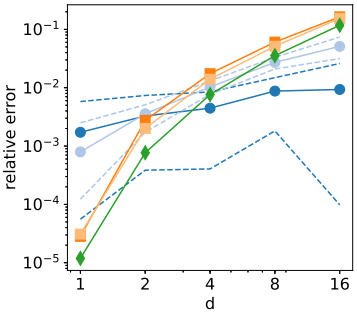
(a) Uniform data, test function 1.



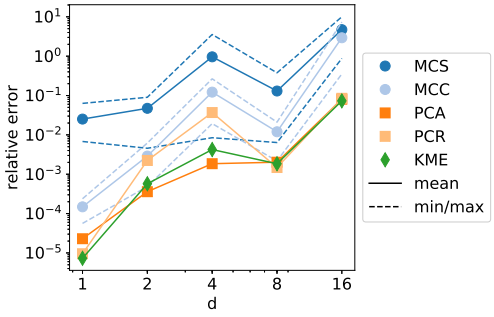
(b) Uniform data, test function 2.



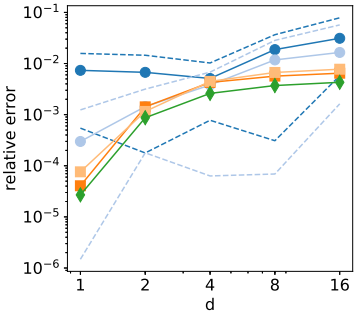
(c) Normal data, test function 1.



(d) Normal data, test function 2.

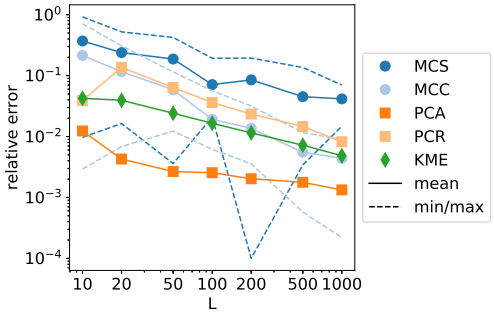


(e) Polynomial data, test function 1.

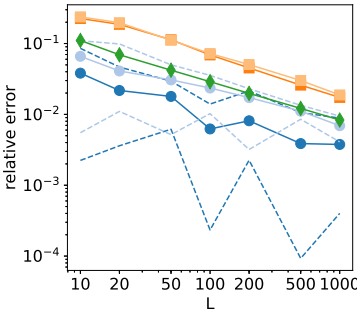


(f) Polynomial data, test function 2.

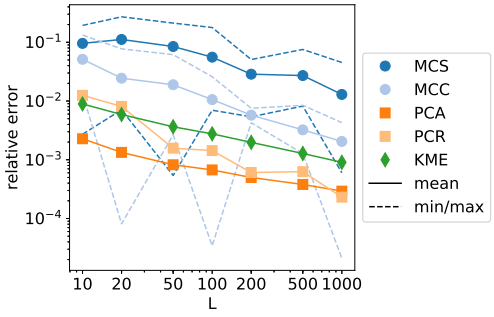
Figure 3.3: Relative error depending on dimension for different methods and data sets with $L = 50$.



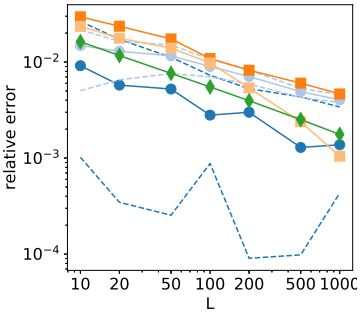
(a) Uniform data, test function 1.



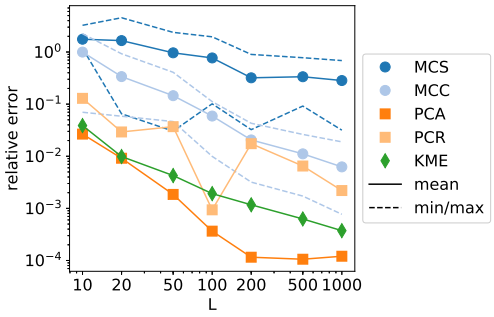
(b) Uniform data, test function 2.



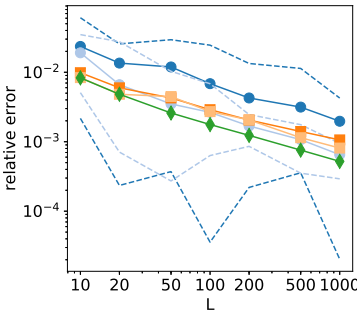
(c) Normal data, test function 1.



(d) Normal data, test function 2.



(e) Polynomial data, test function 1.



(f) Polynomial data, test function 2.

Figure 3.4: Relative error depending on L for different methods and data sets with $d = 4$.

Computational cost

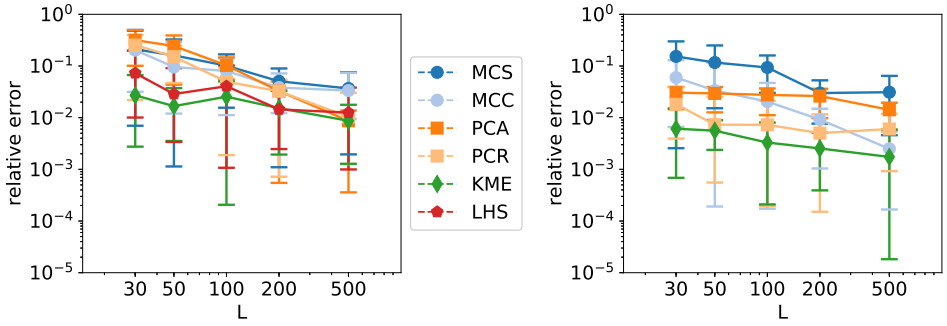
First, we note the time to compute MCS is negligible. The time to perform the clustering is about linear in L and N for all methods, although the constants differ. For the considered methods, PCA is fastest, followed by PCR (about a factor 3 slower for $L = 100$). For MCC, we look at the cost of computing one set of samples instead of ten, as was done to find the effect of randomness. Then, MCC is about as fast as PCR, which is expected because the reassignment step is most costly. Finally, KME (with $r = 10$) is about three to six times slower than MCC. We emphasize this also depends on the implementation of the k-means method; we use the implementation from Scikit-learn [73]. Furthermore, we emphasize that the clustering needs to be carried out only once, as a pre-processing step to determine the nodes (and associated weights) in parameter space at which the expensive model $u(\mathbf{x})$ must be evaluated. For several applications, a single evaluation of $u(\mathbf{x})$ can be orders of magnitude more expensive than the computational cost of the pre-processing to determine the samples.

3.4.4 Ishigami function

We use again the test case described in Section 2.5.1, in which the input data is either uniform on $[-\pi, \pi]^3$ or normally distributed according to (2.11) and range-normalized to $[-\pi, \pi]^3$. The output function is given by (2.10). For this chapter, we compute the integral based on L samples, with $L \in \{30, 50, 100, 200, 500\}$. We use the clustering methods detailed before and add one often-used sampling method, namely Latin hypercube sampling (LHS), which is only applicable for the independent data. In this method, no weights are used. The error measure is again the relative error as given by (3.6). The results are in Figure 3.5. Because of the n_r repetitions of the simulation with different input data sets, we show confidence intervals as well. Here, the Monte Carlo methods have been performed only once. One can see that here KME performs best and reassigning the data points to the nearest cluster does not necessarily improve the result.

3.5 Conclusion

We have proposed a novel collocation method that employs clustering techniques, thereby successfully dealing with the case of multivariate, dependent inputs. We have assessed the performance of this clustering-based collocation method using the Genz test functions as well as the Ishigami function as benchmarks. Three clustering techniques (and one refinement) were considered in this context, namely the Monte Carlo (MCC), k-means (KME) and principal component analysis based (PCA-based and PCR) clustering techniques. Naive Monte Carlo sampling (MCS)



(a) Independent data.

(b) Dependent data.

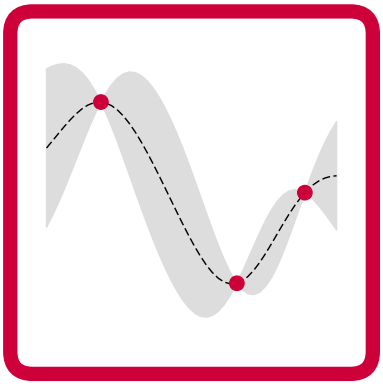
Figure 3.5: Accuracy of the integral computed with different clustering methods.

is used for comparison. No exact knowledge of the input distribution is needed for the clustering-based method proposed here; a sample of input data is sufficient. We hypothesize that the more dependent the inputs are, the more the input data are concentrated on a low-dimensional manifold. This makes it possible to obtain a good representation of the input data with a relatively small number of cluster centers.

In our settings, the variables have similar input ranges. When this is not the case, one needs to nondimensionalize or scale them. We leave this subject open for interpretation of the modeler.

We observed that the nodes obtained with MCC are mostly concentrated in regions of high density of the input probability distribution, with poor representation of the tails. As a result, this method does not perform well for a low number of dimensions. The PCA-based method is better at giving a good spread of the collocation nodes, although its results vary for the different test functions. KME has consistently good results, although it cannot beat Monte Carlo when the data is (nearly) independent. Concerning computational cost, the PCA-based method is fastest. Overall, the computational cost of the clustering methods is small, and will be negligible compared to the computational cost of expensive model evaluations (involving e.g., solvers for computational fluid dynamics).

From the results, we cannot in general advise PCA-based or KME over the other. When a good spread of the collocation nodes over the domain is preferred, PCA-based may be the better choice. On the other hand, when edge effects are not important, KME may be the better choice. In this chapter we have focused on clustering-based quadrature. Collocation is also frequently used as an approach for obtaining approximations of output functions through interpolation or emulation. This subject will be treated in Chapters 4 and 6.



4 Emulation

In the previous chapter we developed a method to select samples, at which the black-box computational model is evaluated. We can use the output generated from these samples for further analysis. However, due to high computational cost, the number of available output samples can be small and this may lead to inaccuracies and large uncertainties in quantities computed from this available output. It is therefore beneficial to obtain more output samples by a surrogate model. In this way, post-processing can be done on the complete input data set after obtaining an emulated output value for each data point.

4.1 Introduction

The challenge which now appears is to use the available output to construct a surrogate model. As the name suggests, a surrogate model (also called an emulator) replaces the complex computational model by a simpler one which is cheaper to evaluate.

The use of surrogate models is wide-spread in engineering and therefore, numerous types exist. They have in common that they are generated from input-output samples only and are in general unaware of the underlying process or model that generated these samples. Among the oldest models are linear regression models and polynomial response surfaces [74]. Later, radial basis functions [75] were used to construct surrogate models. More recently, artificial neural networks [76] came into play. Gaussian process models were first developed in geostatistics [77], but became popular in other research fields as well due to their flexibility and, therefore, wide applicability. In [4], it was proposed to use Gaussian processes for emulation of expensive computational models. A numerical assessment of emulation strategies can be found in [78].

In the following chapters, the concepts of Gaussian processes and Gaussian process regression are extensively used. The first part of this chapter is therefore based on literature and serves as a short introduction to Gaussian processes. The

second part of this chapter contains new results on the derivation of bounds on the prediction variance of a Gaussian process. Furthermore, we show these bounds can be made more strict when k-means clustering is used to obtain the input samples.

Section 4.2 introduces linear regression as preliminary for Section 4.3 which explains Gaussian process regression and illustrates the difference between linear and Gaussian process regression. More details of Gaussian processes can be found in Section 4.4, while Section 4.5 gives useful bounds on the prediction variance.

4.2 Linear regression

Linear regression is one of the easiest ways to construct surrogate models. The surrogate model is defined by a linear combination of a set of regressors, whose parameters are estimated from the available output. Because a regression fit is usually non-interpolating, an error term appears. This section is based on [79, Sections 8.1-8.2]. We choose to include a short description of linear regression because it can help to understand the parameter estimation of the Gaussian process.

We start with the d-dimensional case, where we write

$$u_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_d x_{id} + \varepsilon_i,$$

for each observation $(\mathbf{x}_i, u_i)$ with $i \in \{1, \ldots, L\}$ and $\mathbf{x}_i = (x_{i1}, \ldots, x_{id})$. In this case, the input data is denoted by $\mathbf{x}_i$, while the output of the computational model is denoted by u_i. β_j for $j \in \{0, \ldots, d\}$ are parameters to be estimated and ε_i is the error between u_i and the predicted value $\beta_0 + \beta_1 x_{i1} + \ldots + \beta_d x_{id}$. One can write this surrogate model in matrix form as

$$\mathbf{u} = X\beta + \varepsilon, \quad X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1d} \\ \vdots & \vdots & & \vdots \\ 1 & x_{L1} & \cdots & x_{Ld} \end{bmatrix}, \quad \beta = \begin{bmatrix} \beta_0 \\ \vdots \\ \beta_d \end{bmatrix}, \quad \varepsilon = \begin{bmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_d \end{bmatrix}, \tag{4.1}$$

in which X is called the design matrix. The column vectors of X are called regressors. Note that linear regression does not imply that all regressors are linear functions; it refers to the linear combination of the (possibly nonlinear) regressors. One may also include, for example, the vector x_{i1}^2 in the design matrix X with a corresponding parameter in β.

We will continue with linear regressors only. For this model, some assumptions are used to simplify the estimation of the parameters β. One of them is that the samples are assumed to be given at fixed positions, i.e., there is no uncertainty in the values of $\mathbf{x}_i$. Another assumption is that we have $L > d + 1$.

We consider the general case in which we assume the errors satisfy $\mathbb{E}[\varepsilon|X] = \mathbf{0}$ and $\text{Cov}(\varepsilon|X) = \Sigma$, which means the errors do not need to be independent. When Σ is a diagonal matrix, then the errors are independent. These assumptions are enough to derive the generalized least-squares estimator of β, which is found by minimizing

$$\|\mathbf{u} - X\beta\|_M.$$

Herein, M denotes the Mahalanobis distance with matrix $M = \Sigma$, which is defined as

$$d_M(\mathbf{x}, \mathbf{x}') = \sqrt{(\mathbf{x} - \mathbf{x}')^T M^{-1} (\mathbf{x} - \mathbf{x}')}. \tag{4.2}$$

This leads to the estimator

$$\hat{\beta} = (X^T \Sigma^{-1} X)^{-1} X^T \Sigma \mathbf{u}. \tag{4.3}$$

One can substitute $\Sigma = \sigma^2 I$ to find the ordinary least-squares estimator

$$\hat{\beta} = (X^T X)^{-1} X^T \mathbf{u}.$$

The estimator for the variance of the error, $\widehat{\sigma^2}$, is given by

$$\widehat{\sigma^2} = \frac{\|\mathbf{u} - X\hat{\beta}\|_M^2}{L} = \frac{1}{L}(\mathbf{u} - X\hat{\beta})^T \Sigma^{-1} (\mathbf{u} - X\hat{\beta}), \tag{4.4}$$

which is biased. An unbiased estimator can be obtained by using $L - d - 1$ as denominator. Estimates for $u(\mathbf{x})$ (denoted $m(\mathbf{x})$) can be made by

$$m(\mathbf{x}) = \chi(\mathbf{x})\hat{\beta}, \quad \chi(\mathbf{x}) = \begin{bmatrix} 1 & x_1 & \cdots & x_d \end{bmatrix}. \tag{4.5}$$

When we now assume the errors are (multivariate) normally distributed, we can derive the maximum likelihood estimators. The likelihood is given by

$$\mathscr{L}(\beta, \sigma^2) = \frac{1}{\sqrt{(2\pi)^L \det(\Sigma)}} \exp\left(\frac{-1}{2} (\mathbf{u} - X\beta)^T \Sigma^{-1} (\mathbf{u} - X\beta) \right),$$

leading to the log-likelihood

$$l(\beta, \sigma^2) = \frac{-L}{2} \log(2\pi) - \frac{1}{2} \log(\det(\Sigma)) - \frac{1}{2} (\mathbf{u} - X\beta)^T \Sigma^{-1} (\mathbf{u} - X\beta).$$

The maximum likelihood estimators equal the ones from (4.3) and (4.4).

4.3 Gaussian process regression

One may wonder why we included the maximum likelihood estimators in the previous section, given they (i) need more assumptions and (ii) lead to the same estimators as obtained by least-squares. One usually only obtains the parameter estimates from the least-squares point-of-view. However, we are now going to make the step from linear regression to Gaussian process regression, in which maximum likelihood estimators are used to obtain the parameter estimates. In this section, we will only discuss the general idea. Section 4.4 gives more details on their construction.

We first recall (4.1):

$$\mathbf{u} = X\beta + \varepsilon, \quad X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1d} \\ \vdots & \vdots & & \vdots \\ 1 & x_{L1} & \cdots & x_{Ld} \end{bmatrix}, \quad \beta = \begin{bmatrix} \beta_0 \\ \vdots \\ \beta_d \end{bmatrix}, \quad \varepsilon = \begin{bmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_d \end{bmatrix}.$$

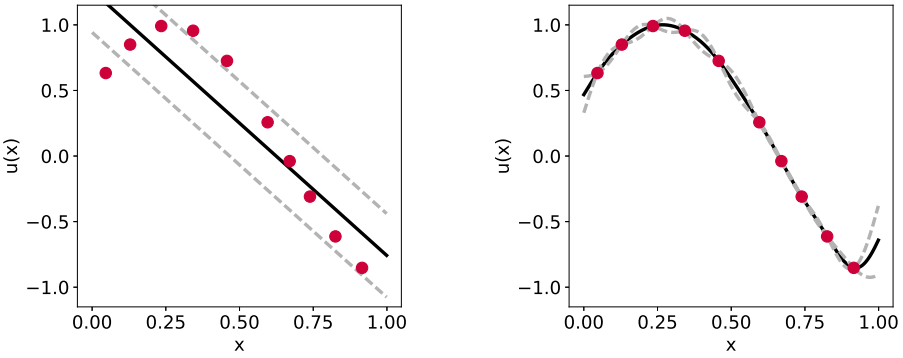
In this expression, the design matrix X can be very extensive, while the errors are very basic. The covariance matrix of these errors equals $\sigma^2 I$. In Gaussian process regression, one only includes a very basic design matrix X, but chooses a very detailed covariance matrix $\Sigma = \sigma^2 K$, in which K is a correlation matrix. The elements of K, denoted K_{ij}, depend on the distance r between observations $\mathbf{x}_i$ and $\mathbf{x}_j$. Due to the construction, the output for unseen values of $\mathbf{x}$ depends on the observations via the correlation function. Unseen values of $\mathbf{x}$ are values which are not used in the regression fit. The advantage of this construction is that the prediction variance is nonconstant, i.e., small when close to observations, and larger when further away from them. More details can be found in Section 4.4.

We also recall (4.5) and use it to write the complex computational model output $u(\mathbf{x})$ for an unseen $\mathbf{x}$ as

$$u(\mathbf{x}) = m(\mathbf{x}) + e(\mathbf{x}) = \chi(\mathbf{x})\hat{\beta} + e(\mathbf{x}).$$

The idea behind the detailed covariance structure in Gaussian process regression is that it can compensate for a very basic design matrix, and the argument is as follows: the chosen regressor functions will in general not model the output $u(\mathbf{x})$ completely. Therefore, there exists a discrepancy (model error) between $m(\mathbf{x})$ and $u(\mathbf{x})$, which one can call $e(\mathbf{x})$. In the case of continuous $u(\mathbf{x})$ and continuous regressors, this means the discrepancy $e(\mathbf{x})$ will be continuous as well. Some values of $\mathbf{x}$ will be under-predicted, while others will be over-predicted. It is not hard to see that values of $\mathbf{x}$ in a small neighborhood of an under-predicted $\mathbf{x}$-value will also be under-predicted due to the continuity, and analogous for over-prediction. Then, it makes sense to correlate them based on the distances between them. Furthermore, due to the interpolation property of Gaussian process regression, one can better capture the behavior of the output.

We illustrate these ideas in Figure 4.1 with a one-dimensional example. The predictors 1 and x are included in the linear regression, while the Gaussian process contains only the predictor 1 and has correlation function $\kappa(r) = \exp\left(\frac{-r^2}{0.1^2}\right)$. The observations obey the relationship $u(x) = \sin(4x + 1/2)$. It is clear that the



(a) Linear regression.

(b) Gaussian process regression.

Figure 4.1: Differences between the regression types. The data points are shown in red, the predictor is shown in black and the predictor plus-minus one σ is shown in gray-dotted lines where appropriate.

Gaussian process captures the (nonlinear) behavior better than the linear regression, as one would expect. Another advantage of the Gaussian process regression can be seen in the varying width of the confidence interval.

4.4 Gaussian processes

Gaussian processes are a generalization of the Gaussian distribution. According to [80],

> A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

This has as a consequence that the joint distribution of any finite number of random variables in a Gaussian process does not depend on the other random variables included therein, which can be infinite in number. This leads to a computationally convenient framework. Also, this leads to consistency, as the distribution of a subset is not changed by the presence of other items in the collection.

A Gaussian process $g(\mathbf{x})$ is completely specified by its mean function ($\mu(\mathbf{x})$) and covariance function ($C(\mathbf{x}, \mathbf{x}')$), and is denoted by

$$g(\mathbf{x}) \sim \mathrm{GP}(\mu(\mathbf{x}), C(\mathbf{x}, \mathbf{x}')).$$

In practice, the domain of the Gaussian process will be $\mathbb{R}^d$ or an open subset of it, with d the dimension of $\mathbf{x}$. This means that for each $\mathbf{x}$ in the domain, the output of the Gaussian process is a normal distribution with a certain mean and variance. The mean and covariance functions $\mu(\mathbf{x})$ and $C(\mathbf{x}, \mathbf{x}')$ do not translate one-to-one to the prediction mean and prediction variance, which are denoted $m(\mathbf{x})$ and $\Sigma(\mathbf{x})$. We will explain later in this section how to obtain $m(\mathbf{x})$ and $\Sigma(\mathbf{x})$. In our setup, the covariance function will depend only on the distance between two locations, and not on the locations themselves (translation-invariant). Furthermore, note that the covariance between two locations is independent of the output value at these locations. The covariance function is modeled by a variance σ^2 multiplied with a kernel function $\kappa(r)$ which serves as a correlation function, in which $r = \|\mathbf{x}-\mathbf{x}'\|$. Several examples of kernel functions are given in Section 4.4.1.

This distance r is often the Mahalanobis distance between $\mathbf{x}$ and $\mathbf{x}'$ (4.2). If $M = I$, then r is the Euclidean distance between $\mathbf{x}$ and $\mathbf{x}'$. If $M = \text{diag}(\mathbf{l})^{-2}$, then r is the Euclidean distance between $\mathbf{x}$ and $\mathbf{x}'$, normalized by the length scales $\mathbf{l}$. We choose to use

$$M = \text{diag}(\mathbf{l})^{-2}, \tag{4.6}$$

such that we use different length scales on different dimensions. These length scales have to be determined as well, which is described in Section 4.4.2. Other distances, such as the L_1-norm, or a scaled version thereof, can be used as well. This may be beneficial if one wants to have a distance which is additive with respect to the different input variables.

Our purpose for using the Gaussian process is to perform regression, also called kriging [77, 80]. Three flavours of kriging are simple kriging, ordinary kriging and universal kriging. The first one assumes a zero mean ($\mu(\mathbf{x}) = 0$), the second a constant (possibly unknown) mean ($\mu(\mathbf{x}) = \mu$), while the third one allows for a polynomial trend as mean ($\mu(\mathbf{x}) = p(\mathbf{x})$). A disadvantage of the first is that prediction far away from measurements leads to prediction values of zero (instead of the mean or something else). We will use the version with the constant mean. One may ask if this is sufficient. The first part of the answer is that linear terms are already captured when the correlation between samples goes to unity, as shown by [81], hence, it is not necessary to explicitly include linear terms. The second part of the answer is that in some cases, simple models perform better than complex models [82]. This means that increasing the complexity of the model is not a guarantee for obtaining better results. Furthermore, [83] shows that the effect of design (choice of samples) is surprisingly large and can be larger than the effect of both the choice of regression component and kernel.

The training set for the regression consists of L observations, denoted by $\mathscr{D} = \{(\mathbf{z}_l, u_l) | l = 1, \ldots, L\} = \{Z, \mathbf{u}\}$. $\mathbf{z}_l$ denotes the input samples, while u_l denotes the computed/measured/observed outputs. We switch to using $\mathbf{z}$ instead of $\mathbf{x}$ to

avoid confusion when writing the set of input samples by a capital letter. The output vector $\mathbf{u}$ is a column vector containing the u_l's, while Z is a matrix of size $L \times d$. The covariance between the observations can be written in a matrix as $C(Z,Z) = \sigma^2 K(Z,Z)$, in which $C_{ij} = C(\mathbf{z}_i, \mathbf{z}_j)$ and $K_{ij} = \kappa(\|\mathbf{z}_i - \mathbf{z}_j\|)$. This leads to the following multivariate normal distribution which is a part of the Gaussian process:

$$g(Z) \sim N\left(\mathbf{1}\mu, C(Z,Z)\right) = N\left(\mathbf{1}\mu, \sigma^2 K(Z,Z)\right),$$

in which the design matrix $X = \mathbf{1}$ of size $L \times 1$ (all entries equal 1, due to the ordinary kriging), whereas μ and σ^2 are yet to be determined. Note that the mean of $g(Z)$ should equal $\mathbf{u}$. This will be accounted for automatically in the computation of the predictions due to the covariance structure. Furthermore, the length scales in $\kappa(r)$ are yet unknown and thereby also the values of $K(Z,Z)$. These are derived in Section 4.4.2.

First, we show how to obtain predictions when these parameters are estimated, indicated by a hat on the variable. Due to the consistency of the Gaussian process (adding observations does not change the distribution of the earlier ones), one can extend this normal distribution with L^* extra input data Z^* as

$$\begin{bmatrix} g(Z) \\ g(Z^*) \end{bmatrix} \sim N\left(\begin{bmatrix} \mathbf{1}_{L\times 1} \\ \mathbf{1}_{L^*\times 1} \end{bmatrix} \hat{\mu}, \widehat{\sigma^2} \begin{bmatrix} K(Z,Z) & K(Z,Z^*) \\ K(Z^*,Z) & K(Z^*,Z^*) \end{bmatrix}\right).$$

From this enlarged distribution, one can now condition on the earlier observations to obtain the (normal) distribution for the new ones, which is a benefit from the multivariate normal distribution. Working this out requires multiple pages of straightforward linear algebra, for which we refer to [84, pp. 428–429]. We just skip to the conclusion:

$$\begin{aligned} g(Z^*) &\sim N\left(m(Z^*), \Sigma(Z^*)\right), \\ m(Z^*) &= \mathbf{1}_{L^*\times 1}\hat{\mu} + K(Z^*,Z)K(Z,Z)^{-1}(\mathbf{u} - \mathbf{1}_{L^*\times 1}\hat{\mu}), \\ \Sigma(Z^*) &= \widehat{\sigma^2}\left(K(Z^*,Z^*) - K(Z^*,Z)K(Z,Z)^{-1}K(Z,Z^*)\right). \end{aligned} \tag{4.7}$$

Hence, given the data $\{Z, \mathbf{u}\}$, we can now make a prediction for a new data point $\mathbf{z}^*$ as a normal distribution with mean $m(\mathbf{z}^*)$ and variance $\Sigma(\mathbf{z}^*)$. Note that one can replace Z^* by a $\mathbf{z}_l \in Z$ to see that the mean of $g(\mathbf{z}_l)$ coincides with u_l and that the variance at that location is zero. Finally, we mention it is numerically easy to obtain these distributions for prediction once $\hat{\mu}$, $\widehat{\sigma^2}$ and $\hat{\mathbf{l}}$ are determined. We continue with several aspects of the implementation of this procedure.

4.4.1 Kernels

The choice of the kernel is important for the behavior of the Gaussian process. The Gaussian process inherits properties of the kernel, like smoothness and length scales. These properties need to be taken into account in choosing a suitable kernel, most importantly the differences between finite and infinite kernels, between smooth and nonsmooth kernels, and between isotropic and anisotropic kernels. Kernel functions $\kappa(r)$ are positive definite functions and have the property $\kappa(0) = 1$. The first requirement guarantees that K will be a covariance matrix (i.e., positive definite for distinct $\mathbf{x}_i$, semi-positive definite otherwise), the second is for scaling purposes.

Note that although a kernel may be differentiable at $r = 0$, if this derivative does not equal 0, the resulting Gaussian process will not be differentiable.

Most often, the squared exponential (Gaussian) kernel,

$$\kappa_G(r) = \exp\left(\frac{-r^2}{2}\right),$$

is used. This kernel is smooth and easy to compute. However, it has infinite support and leads to small values in the covariance matrix, which thereby becomes ill-conditioned.

Therefore, we look for kernels with finite (compact) support, which we require to be stationary (translation-invariant). The difficulty is to find kernels with finite support which are also positive definite. Wendland [85] gives multiple kernel functions $\kappa_{D,q}(r)$ which are polynomials and positive definite up to some maximum input dimension D and have domain $[0,1]$ (nonzero on $[0,1)$). q is a smoothness parameter, in such a way that the first polynomial (with $q = 0$) has a cusp for $r = 0$, while the others have derivative zero for $r = 0$. As q goes to infinity, with proper scaling and linear transformations, the Wendland kernel converges to the Gaussian kernel.

Another often used type of kernels is the family of Matérn kernels [86] defined by

$$\kappa_{M,\nu}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\sqrt{2\nu}d\right)^{\nu} K_\nu\left(\sqrt{2\nu}r\right),$$

where Γ is the gamma function and K_ν the modified Bessel function of the second kind [87]. When $\nu = p + 1/2$, $p \in \mathbb{N}$, this equation reduces to a product of an exponential and a polynomial of order p. A Gaussian process with this kernel is $\lfloor \nu \rfloor$ times differentiable.

Figure 4.2 shows their differences, in which we choose $D = 1$, $q = 1$ and $\nu = 3/2$. The domain of the Wendland kernel has been scaled to $[0,R]$, in which R is optimized to minimize the L_1-norm between $\kappa_{D,q}(r)$ and $\kappa_G(r)$.

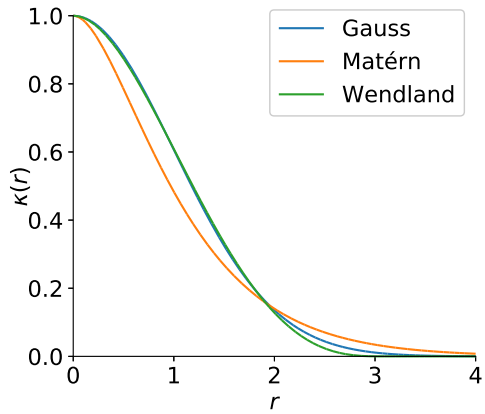


Figure 4.2: The behavior of different kernels. The difference between Gauss and Wendland is negligible in most of the domain, while the Matérn kernel shows a different decay.

4.4.2 Parameter optimization

We continue with the parameter optimization needed to determine $\hat{\mu}$, $\widehat{\sigma^2}$ and $\mathbf{l}$. The set of output observations $\mathbf{u}$ has a multivariate normal distribution with mean $\mathbf{1}\hat{\mu}$ and covariance matrix C with entries $C_{ij} = \sigma^2 K(\mathbf{z}_i, \mathbf{z}_j)$. Hence, its likelihood is given by [88]

$$\mathscr{L}(\mu, \sigma^2, \mathbf{l}|Z, \mathbf{u}) = \frac{1}{\sqrt{(2\pi)^L \det(C)}} \exp\left(-\frac{1}{2}(\mathbf{u}-\mathbf{1}\mu)^T C^{-1}(\mathbf{u}-\mathbf{1}\mu)\right).$$

To emphasize the dependency on the parameters $\mathbf{l}$, we use the notation $K_\mathbf{l}$ instead of K and rewrite the previous expression as

$$\mathscr{L}(\mu, \sigma^2, \mathbf{l}|Z, \mathbf{u}) = \frac{1}{\sqrt{(2\pi\sigma^2)^L \det(K_\mathbf{l})}} \exp\left(-\frac{1}{2\sigma^2}(\mathbf{u}-\mathbf{1}\mu)^T K_\mathbf{l}^{-1}(\mathbf{u}-\mathbf{1}\mu)\right).$$

Taking the logarithm leads to

$$\begin{aligned} l(\mu, \sigma^2, \mathbf{l}|Z, \mathbf{u}) = &-\frac{L}{2}\log(2\pi) - \frac{L}{2}\log(\sigma^2) - \frac{1}{2}\log(\det K_\mathbf{l}) \\ &- \frac{1}{2\sigma^2}\left((\mathbf{u}-\mathbf{1}\mu)^T K_\mathbf{l}^{-1}(\mathbf{u}-\mathbf{1}\mu)\right). \end{aligned} \tag{4.8}$$

In this case, finding the maximum for the parameters leads to a coupled system. Therefore, we are going to solve it in steps. First, we take the derivative of (4.8) with respect to μ, equate this to zero and solve for μ. This gives us (see also [88])

$$\hat{\mu} = \frac{\mathbf{1}^T K_\mathbf{l}^{-1}\mathbf{u}}{\mathbf{1}^T K_\mathbf{l}^{-1}\mathbf{1}}. \tag{4.9}$$

It can be checked via the second derivative that this is indeed a maximum. In the same way, optimizing for σ^2 leads to

$$\widehat{\sigma^2} = \frac{1}{L}\left((\mathbf{u}-\mathbf{1}\hat{\mu})^T K_{\mathbf{l}}^{-1}(\mathbf{u}-\mathbf{1}\hat{\mu})\right), \tag{4.10}$$

in which we use the estimator for $\hat{\mu}$. This expression for σ^2 can be substituted in the log-likelihood to get

$$\begin{aligned} l(\mathbf{l}|Z,\mathbf{u}) = &-\frac{L}{2}\log(2\pi) - \frac{L}{2}\log\left(\frac{1}{L}\left((\mathbf{u}-\mathbf{1}\hat{\mu})^T K_{\mathbf{l}}^{-1}(\mathbf{u}-\mathbf{1}\hat{\mu})\right)\right) \\ &-\frac{1}{2}\log(\det K_{\mathbf{l}}) - \frac{L}{2}. \end{aligned}$$

The constant terms and equal factors can be ignored in the optimization of $\mathbf{l}$, leading to the expression

$$-L\log\left(\frac{1}{L}\left((\mathbf{u}-\mathbf{1}\hat{\mu})^T K_{\mathbf{l}}^{-1}(\mathbf{u}-\mathbf{1}\hat{\mu})\right)\right) - \log(\det K_{\mathbf{l}}), \tag{4.11}$$

which needs to be maximized for $\mathbf{l}$. This can in general not be done analytically and numerical optimization is used. Also, the log-likelihood often has multiple local minima, an example hereof can be found in [80, pp. 115–116]. To avoid local minima with extremely short or long length scales, bounds on the length scales are imposed. Following [89], the bounding box is chosen as $[0.01, 10]^d$.

Now, we can go from estimators to estimates. The numerically optimized value for $\mathbf{l}$ is inserted into (4.9), to obtain the estimate for $\hat{\mu}$. Finally, this estimate is substituted into (4.10).

4.4.3 Numerical stability

In the previous discussion, the numerical implementation has largely been ignored. We will detail here the adaptation used to improve the numerical stability of computing the inverse of $K(Z,Z)$ and the implementation of computing the expression (4.11).

First, we have added a nugget parameter η to the correlation matrix. This implies that instead of the matrix $K(Z,Z)$, we will use $K(Z,Z)+\eta^2 I$, where $\eta = 10^{-4}$. This stabilizes the computation of the inverse. One can question how this term should be interpreted. When one looks at the covariance matrix derived from it, only the diagonal terms representing the variance have changed. When one computes (4.7) with $K(Z,Z)^{-1}$ replaced by $(K(Z,Z)+\eta^2 I)^{-1}$ for a Z^* consisting of one $\mathbf{z}_l \in Z$, one finds that the prediction variance has increased from 0 to $\eta^2\widehat{\sigma^2}$. Therefore, one can interpret this nugget term as a small measurement error.

Second, straightforward computation of $\log(\det K_1)$ leads to problems with near to zero numbers if some samples are near to each other. Therefore, one writes

$$\log(\det(K_1)) = \log\left(\det(U^T U)\right) = \log\left(\det(U)^2\right) = 2\log(\det(U))$$
$$= 2\log\left(\prod_{l=1}^{L} U_{ll}\right) = 2\sum_{l=1}^{L}\log(U_{ll}),$$

and computes this last quantity instead, in which U is the upper-triangular Cholesky factor [90].

4.5 Bounds

We present in this section a new result on the existence and derivation of bounds on the prediction variance. First, it is verified that the prediction variance decreases if more samples are added to the design (as already known). Then, we give a lemma whose assumptions are only met for certain kernels. These assumptions are necessary to derive the bound and therefore limit the number of kernels which are suitable for this theory. These preparations are all given in Section 4.5.1. We derive the full theorem on a bound for the Gaussian process prediction variance in Section 4.5.2. As an extra, we show an adaptation for when the sum-of-squares cannot be split out to different dimensions. Then, we derive the result for isotropic and one-dimensional Gaussian processes as a corollary. As a bonus, we explain why k-means clustering is considered to be optimal for these cases in Section 4.5.3. We finish with the computation of the constant C, which is necessary in the bound, for several kernels in Section 4.5.4.

4.5.1 Notation and preparation

We denote the data set by X, which is a $N \times d$ matrix consisting of N d-dimensional data points $\mathbf{x}$. Z is an $L \times d$ matrix consisting of L samples $\mathbf{z}$ constructed on the data X. Let $\kappa(r)$ denote the chosen (positive definite) kernel and r the Euclidean distance between $\mathbf{x}$ and $\mathbf{x}'$. Let κ^* denote the $L \times 1$ vector $\kappa(Z, \mathbf{x}^*)$ with $\mathbf{x}^*$ a prediction location and K the $L \times L$ matrix $\kappa(Z, Z)$. Let $V(\mathbf{x}^*)$ be the prediction variance at location $\mathbf{x}^*$ of a Gaussian process constructed with kernel $\kappa(r)$ on the training samples Z. Then this prediction variance is given by [91, App. A]

$$V(\mathbf{x}^*) = \widehat{\sigma^2}\left(1 - \kappa^{*T} K^{-1} \kappa^*\right),$$

where $\widehat{\sigma^2}$ is the estimate of the process variance. In what follows, we will initially neglect this factor and look at the normalized variance $V_n(\mathbf{x}^*)$ given by

$$V_n(\mathbf{x}^*) = 1 - \kappa^{*T} K^{-1} \kappa^*. \tag{4.12}$$

We assume the kernel parameters are fixed and $\mathbf{x}^* \notin Z$ for the remainder of this text. We will first give and prove two lemmas which ease the proof of the main result.

Lemma 1. *The normalized prediction variance $V_n(\mathbf{x}^*)$ decreases when an extra sample ζ is added to the set of training samples Z, except when this sample coincides with another sample in Z, thus:*

$$\widetilde{V_n}(\mathbf{x}^*) < V_n(\mathbf{x}^*) \text{ given } \zeta \notin Z,$$

where $\widetilde{V_n}(\mathbf{x}^*)$ denotes the normalized prediction variance after adding ζ.

Proof. Let L be fixed and denote the extra training sample by ζ. Then, the resulting variance estimate can be written as

$$\widetilde{V_n}(\mathbf{x}^*) = 1 - \begin{bmatrix} \kappa^{*T} & \kappa(\mathbf{x}^*, \zeta)^T \end{bmatrix} \begin{bmatrix} K & \kappa(Z,\zeta) \\ \kappa(\zeta, Z) & \kappa(\zeta,\zeta) \end{bmatrix}^{-1} \begin{bmatrix} \kappa^* \\ \kappa(\mathbf{x}^*, \zeta) \end{bmatrix}.$$

Note that $\kappa(\zeta,\zeta) = 1$ and $\kappa(\zeta, Z) = \kappa(Z,\zeta)^T$. We will write ν for $\kappa(Z,\zeta)$ for notational purposes when deemed necessary. It can be proven that the inverse of the extended matrix is given by (see [92]):

$$\begin{bmatrix} K & \nu \\ \nu^T & 1 \end{bmatrix}^{-1} = \frac{1}{1-\nu^T K^{-1}\nu} \begin{bmatrix} (1-\nu^T K^{-1}\nu)K^{-1} + K^{-1}\nu\nu^T K^{-1} & -K^{-1}\nu \\ -\nu^T K^{-1} & 1 \end{bmatrix},$$

when $1 - \nu^T K^{-1}\nu \neq 0$. The corresponding normalized variance is then given by

$$\widetilde{V_n}(\mathbf{x}^*) = 1 - \kappa^{*T} K^{-1} \kappa^* - \frac{(\kappa^{*T} K^{-1}\nu - \kappa(\mathbf{x}^*, \zeta))^2}{1-\nu^T K^{-1}\nu}. \tag{4.13}$$

We see that this is equal to the normalized variance $V_n(\mathbf{x}^*)$ with an extra term. Note that the numerator is a quadratic form, thereby always zero or positive. The denominator can only be zero if $\nu^T K^{-1}\nu = 1$.

We will show i) that the normalized variance $V_n(\mathbf{x}^*)$ (as defined in (4.12)) is only zero for $\mathbf{x}^* \in Z$ (positive otherwise), and from this reasoning, we can show ii) that the denominator in (4.13) will be positive (hence, the expression for the inverse is valid). Finally, we need to show iii) that the numerator in (4.13) is only zero for $\zeta \in Z$.

1. Assume $V_n(\mathbf{x}^*)$ is zero for $\mathbf{x}^*$ and we want to add point $\mathbf{x}^*$ to the GP as training point, so we look at the extended kernel matrix

$$\begin{bmatrix} K & \kappa^* \\ \kappa^{*T} & 1 \end{bmatrix}. \tag{4.14}$$

and replace 1 by the expression $\kappa^{*T} K^{-1} \kappa^*$ (since $V_n(\mathbf{x}^*) = 0$); yielding the kernel matrix

$$\begin{bmatrix} K & \kappa^* \\ \kappa^{*T} & \kappa^{*T} K^{-1} \kappa^* \end{bmatrix}.$$

Then the determinant equals 0. Since K^{-1} exists, the first L columns are independent. Hence, the last column can be written as a linear combination of the previous ones. However, if $\mathbf{x}^* \notin Z$, then the matrix in (4.14) should be positive definite due to the kernel being positive definite! Hence, the only option is when $\mathbf{x}^* \in Z$, where the linear combination can be written as one of the unit vectors in $\mathbb{R}^k$. This was excluded, so $V_n(\mathbf{x}^*) > 0$.

2. With the same reasoning as before, the denominator in (4.13) can only be zero if $\zeta \in Z$, which is excluded since it would not be an extra training sample.

3. It remains to be shown that $(\kappa^{*T} K^{-1} v - \kappa(\mathbf{x}^*, \zeta))^2 = (\kappa(\mathbf{x}^*, Z) K^{-1} \kappa(Z, \zeta) - \kappa(\mathbf{x}^*, \zeta))^2 > 0$ if $\zeta \notin Z$. Assume $(\kappa(\mathbf{x}^*, Z) K^{-1} \kappa(Z, \zeta) - \kappa(\mathbf{x}^*, \zeta))^2 = 0$.

   Consider the extended kernel matrix as before with ζ instead of $\mathbf{x}^*$:

$$\begin{bmatrix} K & v \\ v^T & 1 \end{bmatrix}.$$

   It has reduced rank only when $\zeta \in Z$. We write the matrix

$$\begin{bmatrix} K & v \\ \kappa(\mathbf{x}^*, Z) & \kappa(\mathbf{x}^*, \zeta) \end{bmatrix},$$

   which has a different bottom row than the matrix from before. It turns out that its determinant is again zero (by assumption), while the first L columns are independent. Hence, the last column can be written as a linear combination of the first L ones. Especially, this holds for the first L rows of the last column. Due to the positive definiteness, we can conclude that $\zeta \in Z$, so for $\zeta \notin Z$, we have $(\kappa(\mathbf{x}^*, Z) K^{-1} \kappa(Z, \zeta) - \kappa(\mathbf{x}^*, \zeta))^2 > 0$.

Combining these results, we first see that the last term of (4.13) (including the minus sign) is always negative for $\zeta \notin Z$. From this, we get

$$\widetilde{V_n}(\mathbf{x}^*) < V_n(\mathbf{x}^*),$$

for $\mathbf{x}^* \notin Z$ and $\zeta \notin Z$. □

Basically, this result tells us that adding samples will reduce the prediction variance (under the condition that the Gaussian process parameters are not refit). Therefore, adding samples does not deteriorate the predictions made with

the Gaussian process. The next lemma helps us to determine which kernels are suitable to bound the prediction variance by the sum-of-squares of the design.

Lemma 2. *Assume $\kappa(r)$ is a kernel function with $\kappa(r=0)=1$ and $\kappa(r)$ is two times differentiable on $[0,1)$. If*

$$\kappa'(0)=0,$$

and

$$\exists C_1 \in \mathbb{R} > 0 : |\kappa''(r)| \leq C_1 \text{ for all } r \in (0,1),$$

then,

$$1-\kappa(r)^2 \leq C \cdot r^2 \text{ for all } r \geq 0,$$

with $C = \max\{1, C_1\}$.

Proof. Since $|\kappa(r)| < 1$ for $r \neq 0$ [93], we have automatically $1-\kappa(r)^2 \leq r^2$ for $r \geq 1$. For $r = 0$, this holds as well since $\kappa(0) = 1$. So, we assume for now $r \in (0,1)$. We use the Taylor expansion with remainder

$$\kappa(r) = \kappa(0) + \kappa'(0)r + (1/2)\kappa''(s)r^2,$$

for some $s \in (0,r)$. Then,

$$\begin{aligned}
1-\kappa(r)^2 &= 1-(\kappa(0)+\kappa'(0)r+(1/2)\kappa''(s)r^2)^2, \\
&= 1-(\kappa(0)^2+2\kappa(0)\kappa'(0)r+(\kappa(0)\kappa''(s)+\kappa'(0)^2)r^2+\kappa'(0)\kappa''(s)r^3 \\
&\quad +(1/4)\kappa''(s)^2r^4), \\
&= -2\kappa'(0)r-(\kappa''(s)+\kappa'(0)^2)r^2-\kappa'(0)\kappa''(s)r^3-(1/4)\kappa''(s)^2r^4.
\end{aligned}$$

We now use the assumptions and find

$$1-\kappa(r)^2 = -\kappa''(s)r^2-(1/4)\kappa''(s)^2r^4 \leq -\kappa''(s)r^2 \leq C_1 \cdot r^2.$$

Combining the results for $r=0$, $r \in (0,1)$ and $r \geq 1$ finishes the proof. □

This lemma is important as it indicates the requirement that kernels have to meet in order for our result to hold. For a given kernel, it is easy to evaluate whether the assumptions of Lemma 2 hold. The computation of C will be detailed in Section 4.5.4.

Finally, we give some definitions which will be used in the theorem and the proof. Herein, $\mathbf{z}_i$ denotes the design point corresponding to data point $\mathbf{x}_i$. The sum-of-squares for the input variable j is given by

$$SOS_j = \sum_{i=1}^{N}(x_{ij}-z_{ij})^2,$$

and the sum-of-squares in the input space between the design and the data set is given by

$$SOS = \sum_{j=1}^{d} SOS_j.$$

4.5.2 Main theorem

Theorem 3. *Let X be the data set and Z the design. Let $V_n(\mathbf{x})$ be the normalized prediction variance for a Gaussian process with optimized length scales $\mathbf{l}$ constructed on the design Z with chosen kernel $\kappa(r)$ for which Lemma 2 holds with normalized Euclidean distance r (Mahalanobis distance with M diagonal and $M_{ii} = 1/l_i^2$). Denote by SOS the sum-of-squares as defined before. Then, there exists a $C > 0$ such that*

$$\sum_{i=1}^{N} V_n(\mathbf{x}_i) \leq C \sum_{j=1}^{d} \frac{SOS_j}{l_j^2}. \tag{4.15}$$

Proof. Define a piece-wise normalized Gaussian process by local Gaussian processes $\overline{V}_n(\mathbf{x}_i)$ on each of the Voronoi cells defined by Z and the normalized Euclidean distance r. Each local Gaussian process is built on the training sample in that Voronoi cell with kernel $\kappa(r)$, distance r and process variance 1. Therefore, for each $\mathbf{x}_i$ only the nearest $\mathbf{z}_j$ is taken into account in the piece-wise Gaussian process. We denote by $\tilde{Z}$ the $N \times d$ matrix consisting of N row vectors $\tilde{\mathbf{z}}_i$, such that $\tilde{\mathbf{z}}_i$ is the nearest $\mathbf{z}_j$ for $\mathbf{x}_i$. Then, we get

$$\begin{aligned}
\sum_{i=1}^{N} V_n(\mathbf{x}_i) \leq \sum_{i=1}^{N} \overline{V_n}(\mathbf{x}_i) &= \sum_{i=1}^{N} \left(1 - \kappa(\|\mathbf{x}_i - \tilde{\mathbf{z}}_i\|_2)\kappa(\|\tilde{\mathbf{z}}_i - \tilde{\mathbf{z}}_i\|_2)^{-1}\kappa(\|\mathbf{x}_i - \tilde{\mathbf{z}}_i\|_2)\right), \\
&= \sum_{i=1}^{N} \left(1 - \kappa(\|\mathbf{x}_i - \tilde{\mathbf{z}}_i\|_2)^2\right) = \sum_{i=1}^{N} \left(1 - \kappa(r_i)^2\right), \\
&\leq \sum_{i=1}^{N} C r_i^2 = C \sum_{i=1}^{N} \sum_{j=1}^{d} \frac{(x_{ij} - \widetilde{z_{ij}})^2}{l_j^2} = C \sum_{j=1}^{d} \frac{SOS_j}{l_j^2}.
\end{aligned}$$

The first inequality used Lemma 1, the second one Lemma 2 and we recalled the definition of r. □

We have now shown the normalized prediction variance can be bounded by the sum-of-squares in each dimension. We can generalize this to a bound based on the total sum-of-squares, which will be less tight though. The resulting bound, which equals a factor times the sum-of-squares, is given in Corollary 4.

Corollary 4. *Let the requirements for Theorem 3 be fulfilled. Then, there exists a* $C > 0$ *such that*

$$\sum_{i=1}^{N} V_n(\mathbf{x}_i) \leq \frac{C}{\min_j l_j^2} SOS. \tag{4.16}$$

Proof. The proof follows the one of Theorem 3, with the extra step

$$\sum_{i=1}^{N} V_n(\mathbf{x}_i) \leq C \cdot \sum_{j=1}^{d} \frac{SOS_j}{l_j^2} \leq \frac{C}{\min_j l_j^2} SOS.$$

This holds because all $l_j \neq 0$. □

The obtained results concern d-dimensional anisotropic Gaussian processes, which means d length scales are involved. We give the results for d-dimensional isotropic and one-dimensional Gaussian processes in the following corollary.

Corollary 5. *Let the requirements for Theorem 3 be fulfilled with either* $\mathbf{l} = l\mathbf{1}$*, i.e.,* $l_j = l$ *for each dimension* j *or* $\mathbf{l} = l$*, i.e.,* $d = 1$*. Then*

$$\sum_{i=1}^{N} V_n(\mathbf{x}_i) \leq \frac{C}{l^2} SOS. \tag{4.17}$$

Proof. The proof follows directly from Theorem 3 or Corollary 4. □

These results are independent of the choice of design, i.e., they hold for Monte Carlo sampling as well as for advanced design methods. Therefore, we show in Section 4.5.3 how these results can be used. We end this subsection with some remarks.

We note that we use the normalized prediction variance instead of the "regular" prediction variance. It is necessary to have an estimate $\widehat{\sigma^2}$ in order to translate this result to the "regular" prediction variance. This estimate is available if the bound is used after evaluating all samples. In this case, the advantage of the bound is not having to compute the prediction variance for each data point individually. If the estimate is obtained from an initial design or earlier results, then the advantage of the bound is also to be able to bound the prediction variance before evaluation of the new samples. This is only useful, however, if $\widehat{\sigma^2}$ is not expected to change considerably if the new samples are added.

In practice, the output is nondimensionalized to avoid numerical issues in the scaling of the Gaussian process, which results in values of $\widehat{\sigma^2}$ between 0 and 100. This corresponds to the standard deviation of a prediction being between 0 and 10, which is one order of magnitude in the worst case. Also, a range of values for

$\widehat{\sigma^2}$ may be estimated from earlier experiments, of which the worst case can again be used.

A question which can rise here is whether it is beneficial to refit the Gaussian process after obtaining the new samples. On one hand, this might lead to better estimates of $\widehat{\sigma^2}$ and $\mathbf{l}$, and thereby better predictions. On the other hand, the old bound is not guaranteed if the estimates change.

4.5.3 Optimality of k-means

Inequalities (4.15), (4.16) and (4.17) show that the bound will decrease if the sum-of-squares (SOS) between the design and the data set is decreased. This is exactly the minimization criterion in the k-means clustering. The similarity is caused by the distance function in the Gaussian process being the Euclidean distance (multiplied with a constant factor) instead of the normalized Euclidean distance.

Therefore, using k-means to construct the sampling design is optimal to obtain a low SOS in the case of an isotropic or one-dimensional Gaussian process. A low SOS is beneficial as it represents a low prediction variance, which means the Gaussian process gives small confidence intervals when predicting the output for unseen input data values. The optimality of the k-means clustering is beneficial as it is an easily applicable method to compute the sample points in practice.

The case of a d-dimensional anisotropic Gaussian process is more complicated because of the multiple length scales involved. These are not known when the sampling design is constructed, and therefore, the normalized Euclidean distance cannot be used as distance function in the k-means clustering to optimize the result and thereby minimize the sum-of-squares. However, using k-means will still lead to a small overall SOS and therefore it is still a good heuristic approach to use a k-means sampling design.

4.5.4 The constant C

We now give more details on the computation of the constant C. We compute here the values of C for the Gauss kernel and the Matérn kernel with $\nu = 3/2$, as the bound does not hold for $\nu = 1/2$ due to the cusp for $r = 0$. For Gauss, we get

$$\left|\frac{d^2\kappa_G(r)}{dr^2}\right| = \left|\frac{d^2}{dr^2}\left(\exp\left(\frac{-r^2}{2}\right)\right)\right| = \left|(r^2-1)\exp\left(\frac{-r^2}{2}\right)\right| \leq |r^2-1| \leq 1,$$

hence $C = 1$.

For the Matérn kernel with $\nu = 3/2$, we can simplify the kernel expression to

$$\kappa_{M,\nu=3/2}(r) = (1+\sqrt{3}r)\exp(-\sqrt{3}r).$$

Then, we find the bound

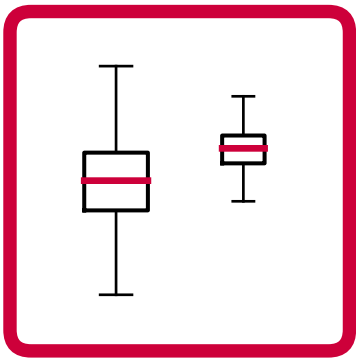
$$\left|\frac{d^2\kappa_{M,\nu=3/2}(r)}{dr^2}\right| = \left|\frac{d^2}{dr^2}\left((1+\sqrt{3}r)\exp(-\sqrt{3}r)\right)\right| = \left|3(r\sqrt{3}-1)\exp\left(-\sqrt{3}r\right)\right|$$
$$\leq \left|3(r\sqrt{3}-1)\right| \leq 3,$$

in which we used $r \in (0,1)$, hence $C = 3$.

For the Wendland kernel, we tabulate some of the coefficients in Table 4.1 for the non-optimized definition, i.e., $\kappa_W(r=1) = 0$. They are computed in a similar way as for the Matérn kernels. However, since they are polynomials on $[0,1]$, they are their own Taylor series on $[0,1]$. Note that the values for C here are higher. This has to do with the Wendland kernels being zero at $r = 1$, so they decay faster. Note that in this case also the estimated length scales will be larger and therefore diminish the effect of a larger C. In the implementation, we choose to include the optimized version of the scaling factor R in the kernel definition, such that the resulting length scales are comparable between the different kernels. This has no effect on the application of Theorem 3, because R can also be regarded as included in the distance function.

Table 4.1: Values of C for the Wendland kernels $W_{d,q}$. $q = 0$ is left out because no bounds exist.

	$q=1$	$q=2$
$d=1$	12	14
$d=3$	20	18 2/3
$d=5$	30	24



5 Sensitivity analysis

The goal of this chapter is to develop a sensitivity analysis method, continuing the line of research started in Chapter 2. We develop two types of sensitivity indices and estimate them via different methods.

5.1 Introduction

Sensitivity analysis is an essential part of uncertainty quantification and a very active research field [94–96]. Several types of sensitivity indices have been formulated, such as variance-based (including Sobol's indices [97]), density-based [98], derivative-based [99] or divergence-based. Broadly speaking, divergence-based sensitivity indices quantify the difference between the joint probability distribution (or density) of model input and output on the one hand, and the product of their marginal distributions on the other hand. A variety of divergence-based indices can be brought together in a common framework built on the notion of f-divergence [100], as was shown by Da Veiga [101]. The f-divergence is a generalization of several well-known divergences such as the Kullback-Leibler divergence [30] and the Hellinger distance [38].

In most cases, these sensitivity indices cannot be computed analytically because the distribution of the model output given the input is not known exactly. As an alternative, one can resort to Monte Carlo sampling (MCS) combined with kernel density estimation (KDE): the input distribution is sampled using MCS, the model is evaluated on all sampled input points, and from resulting input-output points the joint and marginal probability densities of input and output are estimated. However, when the number of available output points is low, for example because of high computational cost of the model, the estimated densities will generally be inaccurate, resulting in large errors in the estimated sensitivity indices.

Mainly based on A. Eggels, and D. Crommelin, "Efficient estimation of divergence-based sensitivity indices with Gaussian process surrogates," *in preparation*, 2019.

In this study we propose to increase the number of output samples by using a Gaussian process (GP) surrogate. The GP is constructed on the input-output points that are obtained with the expensive model. The main idea is that the additional output samples improve the estimation of sensitivity indices even though they introduce a bias due to the difference between the true model and its GP approximation.

Our approach is based on both the development of divergence-based indices and the use of Gaussian processes in sensitivity analysis. Therefore, we briefly summarize some of the advancements in these areas. Auder & Iooss [28] presented two sensitivity analysis methods based on Shannon and Kullback-Leiber entropy, respectively, building on work in [102] and [29]. Da Veiga [101] introduced sensitivity indices based on the f-divergence. In [103], besides more theoretical results, three approximate methods are discussed: one using MCS, one with MCS combined with kernel density estimation (KDE-MC), and one in which MCS, KDE and polynomial dimensional decomposition (PDD-KDE-MC) are combined (see also [104]). Recently, KDE also appears in estimators of mutual information measures [32], where f-divergences are computed between the joint distribution of two random variables and the product of their marginal distributions. Another application of KDE can be found in [105] where a general consistency result for given-data estimation of several sensitivity measures is derived. In [41], f-divergence measures are computed by a k-nearest neighbor graph.

The use of GPs is discussed in Marrel et al. [106], together with the analytical expressions for Sobol indices that arise from them. To compute the indices, two approaches are considered: one in which the predictor of the GP is used and one in which the full GP is used. The latter approach is found to be superior in convergence and robustness. Furthermore, the modeling error of the GP is integrated through confidence intervals; it is reported that the bias due to the use of the GP is negligible [106]. In a related study, Svenson et al. [107] estimate Sobol indices with GPs, using specific compactly supported kernel functions. Furthermore, combining GPs with derivative-based indices has been investigated by [99] and [108]. In [109], predictions from a GP are used to rank the input variables based on their predictive relevance. Two methods for this are presented therein: one based on Kullback-Leibler divergence and one based on the variance of the posterior mean.

Despite the developments sketched above, approaches that combine GP surrogate modeling and divergence-based sensitivity analysis have not been explored much yet (an exception is [110]). The methodology proposed in this chapter combines these two elements. We discuss two variants of this method, one in which only the GP mean is used, and one which also accounts for the GP prediction variance. Both use the KDE method for estimation. For a specific choice of divergence (Hellinger distance), another estimation method based on minimum

spanning trees (see Chapter 2) is proposed as well. This because the sensitivity indices which will be derived in Section 5.2.1 can also be computed by the minimum spanning tree (MST) approach, as detailed in Section 5.2.3.

We note that for the approaches proposed here it is not needed to assume that the inputs are mutually independent, nor does dependency of inputs make it more complicated. We present a test case with independent inputs as well as a case with dependent inputs. For the former, we compare with results obtained with PDD-KDE-MC [103, 104]. PDD-KDE-MC follows a similar philosophy of using surrogates to enlarge the set of points used for density estimation, although the type of surrogate differs. However, the PDD is connected to the analysis-of-variance representation of a multivariate function, which becomes cumbersome in the case of a dependent input distribution.

Section 5.2 describes the sensitivity indices central to this paper, the complications of estimating them and our proposed methods. In Section 5.3, these estimators are applied to three test cases. A second type of sensitivity indices is developed in Section 5.4. Section 5.5 concludes.

5.2 Divergence-based sensitivity indices and their estimation ⋆

5.2.1 Sensitivity indices from the f-divergence

We consider the situation where a model takes a matrix consisting of one input vector $\mathbf{x}_i$ of length d per data point ($X = (\mathbf{x}_1, \dots, \mathbf{x}_N)^T$) and returns an output matrix Y. The input matrix X is random, and as a result the output Y is random as well. Depending on the number of outputs, Y is either a vector or a matrix. Da Veiga [101] proposed to perform global sensitivity analysis with dependence measures, especially f-divergences (see also [103]). In this way, the impact of the kth input variable X^k on the output Y is given by

$$S_{X^k} = \mathbb{E}\left[d(Y, Y|X^k)\right], \tag{5.1}$$

where $d(\cdot,\cdot)$ denotes a dissimilarity measure. The unnormalized first-order Sobol indices can also be written in this framework, namely with

$$d(Y, Y|X^k) = \left(\mathbb{E}(Y) - \mathbb{E}(Y|X^k)\right)^2.$$

We will use the Csiszár f-divergence [100], which is given by

$$d_f(Y, Y|X^k) = \int_{\mathbb{R}} f\left(\frac{p_Y(y)}{p_{Y|X^k}(y)}\right) p_{Y|X^k}(y)\mathrm{d}y, \tag{5.2}$$

with $f(\cdot)$ a convex function with $f(1) = 0$, and $p_{\cdot}(\cdot)$ denoting a probability density function. Hence, p_Y represents the probability density function of the output Y, $p_{Y|X^k}$ the one for Y given X^k and $p_{X^k,Y}$ denotes the joint probability density function of X^k and Y. Some well-known choices for f are $f(t) = -\log(t)$ (Kullback-Leibler divergence) and $f(t) = (\sqrt{t}-1)^2$ (Hellinger distance). Combining (5.1) and (5.2) with basic probability theory gives us

$$S^f_{X^k} = \int\int_{\mathbb{R}^2} f\left(\frac{p_{X^k}(x)p_Y(y)}{p_{X^k,Y}(x,y)}\right) p_{X^k,Y}(x,y)\mathrm{d}y\mathrm{d}x. \tag{5.3}$$

These sensitivity indices are equal to zero for X^k and Y independent and positive otherwise. Furthermore, they are invariant with respect to smooth and uniquely invertible transformations of X^k and Y [111], in contrast to Sobol indices which are only invariant with respect to linear transformations. Moreover, it is easy to generalize (5.3) to multi-dimensional $X^{k,l}$.

5.2.2 Difficulties for estimation

The main problem for computing $S^f_{X^k}$ is that the probability densities in (5.3) are not known. In order to estimate $S^f_{X^k}$ it is necessary to estimate $p_Y(\cdot)$ and $p_{X^k,Y}(\cdot,\cdot)$, and, depending on the type of input, $p_{X^k}(\cdot)$ as well. In [101] it is indicated that if L samples (X_L, Y_L) are available, only the ratio $r(x,y) = \frac{p_Y(y)p_{X^k}(x)}{p_{X^k,Y}(x,y)}$ needs to be estimated.

The estimates of the densities can be obtained with kernel-density estimation (see also [101, 103]). To do so, one chooses a suitable kernel and a suitable value for the kernel bandwidth h, for which guidelines are available [45].

Clearly, the estimate of the density p_Y will not be perfect, leading to an error in the estimation of $S^f_{X^k}$. This is strongly related to the number of samples (X_L, Y_L) available for density estimation. If high computational cost of the model limits this number, the estimation of $S^f_{X^k}$ can be improved by using a surrogate of the model to generate more samples. An existing method for this is polynomial dimensional decomposition (PDD) [112]. It is based on the assumption of mutually independent input variables, which can be unrealistic in cases obtained from practice. Furthermore, the surrogate modeled by PDD is a polynomial, thereby limiting the output function space. Another point of interest is the large number of parameters which needs to be fit for PDD.

As an alternative, we propose to use Gaussian processes [80] as a surrogate model to obtain the larger sample $(X_+, Y_+) = (X_L \cup X_{L^+}, Y_L \cup Y_{L^+})$, in which Y_{L^+} indicates the surrogate model output for the extra input samples X_{L^+}. For each data point in X_{L^+}, this Y_{L^+} is a normal distribution in itself, and for each point in X_L

it is a degenerated normal distribution (i.e., it has zero variance). An additional advantage may be the availability of confidence intervals for $S^f_{X^k}$ at almost no extra computational cost. We note that these confidence intervals do not include the bias from approximating the output by a Gaussian process.

5.2.3 Estimation using Gaussian processes

We assume the input samples $X_L := \{\mathbf{x}_l\}_{l=1}^L$ are already available, otherwise one can use Monte Carlo sampling (or Latin hypercube sampling in the case of independent uniform data) to select samples from the data X. Although it may be tempting to use k-means clustering or other sample selection methods, it is not guaranteed that they represent the distribution just as naive sampling would. Then, the corresponding output $Y_L := \{y_l\}_{l=1}^L$ can be obtained as $Y_L = G(X_L)$ with G the process to generate output, which is either a function or a computational model. Then, one needs to fit a Gaussian process $\widetilde{G}_{\{X_L,Y_L\}}(\mathbf{x}) = N(\mu(\mathbf{x}), \Sigma(\mathbf{x}))$ to (X_L, Y_L), thereby choosing an appropriate kernel. This Gaussian process is now used to obtain output $Y_{L^+} = \widetilde{G}_{\{X_L,Y_L\}}(X_{L^+})$ for other input samples X_{L^+}. This leads to the augmented data set $X_+ = X_L \cup X_{L^+}$ of size $N = L + L_+$ with (partial) surrogate output $Y_+ = Y_L \cup Y_{L^+}$. Note that Y_{L^+} does not consist of single values, but rather of multivariate normal distributions.

Kernel density estimation

We now explain how to compute the KDE on (X_+, Y_+) and how it is used to approximate (5.3). Because $S^f_{X^k}$ is computed per input variable X^k, it is here enough to consider one-dimensional kernel densities.

For each input variable X^k and output variable Y, the estimators for the kernel density are given by [103]:

$$\widehat{f_{X^k}}(x) = \frac{1}{Jh_{X^k}} \sum_{j=1}^{J} K_{X^k}\left(\frac{x - x_j}{h_{X^k}}\right),$$

$$\widehat{f_Y}(y) = \frac{1}{Jh_Y} \sum_{j=1}^{J} K_Y\left(\frac{y - y_j}{h_Y}\right),$$

$$\widehat{f_{X^k,Y}}(x,y) = \frac{1}{Jh_{X^k}h_Y} \sum_{j=1}^{J} K_{X^k}\left(\frac{x - x_j}{h_{X^k}}\right) K_Y\left(\frac{y - y_j}{h_Y}\right),$$

with (x_j, y_j) the jth sample of the input data (X^k, Y) and J the size of the data. Note the input data $X = (X^1, \ldots, X^d)$ has to represent the distribution of X. An extension to a higher-dimensional X^k is easy to obtain. For our purpose, we either

have $J = L$ and $(X, Y) = (X_L, Y_L)$, or we have $J = N$ and $(X, Y) = (X_+, Y_+)$. We choose the Gaussian kernel and $h_{X^k} = h_Y = h$ according to Scott's rule [45], which is optimized with respect to the normal distribution. Then, the estimator for $S^f_{X^k}$ as given by [103] is obtained:

$$\overline{H}^{(J)}_{X^k,f} := \frac{1}{J}\sum_{j=1}^{J} f\left(\frac{\widehat{f_{X^k}}(x_j)\widehat{f_Y}(y_j)}{\widehat{f_{X^k,Y}}(x_j, y_j)}\right). \tag{5.4}$$

We note this choice of h may not be optimal. We have adapted the bandwidth h previously to the ranges of X and Y, but the results of this are worse than with a single bandwidth. Also, kernel density estimation may not be the best choice when the domain of a variable X^k or Y is bounded and this variable has nonzero density at the boundaries.

Until so far, we ignored the fact Y_{L^+} is a multivariate normal random variable instead of a single value when $J = N$. Therefore, there are two options to obtain values for Y_{L^+}. The first option is to use the prediction mean $\mu(x)$ and get the resulting output samples

$$Y_{L^+} = \mu(X_{L^+}), \tag{5.5}$$

to be used in (5.4). The other is to sample from this normal distribution n_s times. In that case, one gets the n_s output sets

$$Y^{(s)}_{L^+} \sim N(\mu(X_{L^+}), \Sigma(X_{L^+})), \tag{5.6}$$

in which $\sim$ denotes "sampled from the distribution", and thereby n_s estimates of $\overline{H}^{(N)}_{X^k,f}$. Note that this also implies the kernel density estimates have to be computed n_s times.

Minimum spanning trees

We can relate the proposed sensitivity indices to the Rényi divergence $D_{1/2}$ as introduced in Chapter 2 in case the used divergence is the Hellinger distance. As briefly mentioned in Section 2.2.2, the Hellinger distance can be computed from $D_{1/2}$ by straightforward transformations. We show the relation between $D_{1/2}$ and $S^H_{X^k}$ here, starting with (2.3), in which $\alpha = 1/2$ is substituted and the dummy variables are numbered:

$$D_{1/2}(g_1, g_2) = -2\log\left(\int_\Omega \sqrt{\frac{g_1(x)}{g_2(x)}}g_2(x)\mathrm{d}x\right) = -2\log\left(\int_\Omega \sqrt{g_1(x)g_2(x)}\mathrm{d}x\right).$$

In our case, g_1 is given by $p_{X^k,Y}$ and g_2 is given by $p_{X^k}p_Y$, hence

$$D_{1/2}(p_{X^k,Y}, p_{X^k}p_Y) = -2\log\left(\int_{\Omega_{X^k}}\int_{\Omega_Y} \sqrt{p_{X^k,Y}(x,y)p_{X^k}(x)p_Y(y)}\mathrm{d}y\mathrm{d}x\right). \tag{5.7}$$

On the other hand, $S^H_{X^k}$, with H denoting the sensitivity index derived from the Hellinger distance, is given through (5.3) by

$$S^H_{X^k} = \iint_{\mathbb{R}^2} \left(\sqrt{\frac{p_{X^k}(x)p_Y(y)}{p_{X^k,Y}(x,y)}} - 1 \right)^2 p_{X^k,Y}(x,y)\mathrm{d}y\mathrm{d}x,$$

which can be simplified to

$$S^H_{X^k} = 2 - 2\iint_{\mathbb{R}^2} \sqrt{p_{X^k}(x)p_Y(y)p_{X^k,Y}(x,y)}\mathrm{d}y\mathrm{d}x. \tag{5.8}$$

We now see the agreement between (5.7) and (5.8). In case the domain of X^k and Y is extended to $\mathbb{R}$ by zero density outside of the domain, it is possible to write

$$D_{1/2}(p_{X^k,Y}, p_{X^k}p_Y) = -2\log(I), \quad S^H_{X^k} = 2 - 2I,$$

with

$$I = \iint_{\mathbb{R}^2} \sqrt{p_{X^k}(x)p_Y(y)p_{X^k,Y}(x,y)}\mathrm{d}y\mathrm{d}x,$$

hence

$$S^H_{X^k} = 2 - 2\exp\left(\frac{-D_{1/2}(p_{X^k,Y}, p_{X^k}p_Y)}{2}\right).$$

We can compute $S^H_{X^k}$ via $D_{1/2}(p_{X^k,Y}, p_{X^k}p_Y) = -H_{1/2}(h)$ (Equation 2.5). Therefore, we need to estimate $L_\gamma(X^k, Y)$ (2.7) and β (2.8), where β is obtained from 10^3 replicates for $N = 10^3$. Because I can be estimated as

$$\frac{L_\gamma(X^k, Y)}{\beta\sqrt{N}},$$

we estimate the sensitivity indices by

$$\widehat{S^H_{X^k}} = 2 - 2\frac{L_\gamma}{\beta\sqrt{N}}. \tag{5.9}$$

5.3 Results

We test the estimators in several ways. The first test case is with regard to random input/output data and is described in Section 5.3.1. In this case, the estimates should be near zero. The second test case compares the results of the estimators in a setting where the exact value of $S^H_{X^1}$ is known (Section 5.3.2). The third test case is based on the Ishigami function and is performed for both independent and

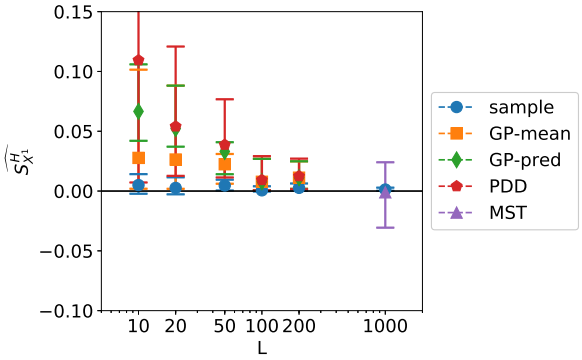


Figure 5.1: Sensitivity indices for random output.

dependent input data, of which the results can be found in Section 5.3.3. The results are summarized in Section 5.3.4.

We use $f(t) = (\sqrt{t}-1)^2$ (Hellinger distance) to include the estimator based on minimum spanning trees. We compare several estimation methods of $\widehat{S^H_{X^k}}$. First, we estimate $\widehat{S^H_{X^k}}$ by kernel density estimation on the L samples only, which is referred to as "sample" in the figures. The other estimation methods make use of the emulator. We also use kernel density estimation with the emulated output samples given by (5.5), which is referred to as "GP-mean", and kernel density estimation with the emulated output samples given by (5.6), which is referred to as "GP-pred" and for which we choose $n_s = 10$. Emulated output as given by (5.5) is also used for the minimum spanning tree method (5.9), which is denoted by "MST". The PDD-KDE-MC method [103], denoted "PDD", is included for comparison, where univariate decomposition ($S = 1$) is used with polynomial order $m = 4$ for both random data and the Ishigami function.

All experiments have been performed $n_r = 10$ times with $n_s = 10$ samples in the case of the estimator "GP-pred". The error bars in the upcoming figures indicate the minimum and maximum obtained value, which are used as an approximation of the 95% confidence interval.

5.3.1 Random data

First, we check the behavior for random output, i.e., independent of the input, in which case the sensitivity indices should be zero. Both the input and output data are one-dimensional, uniformly distributed on $[0,1]$ and have size $N = 10^3$, while L is varied from $L = 10$ to $L = 200$. The experiment is repeated $n_r = 10$ times. The results are in Figure 5.1. On the right, we show the sensitivity index as computed on the complete, i.e., $L = N$, data (blue circle) with the kernel density

estimate. As expected, their mean is around zero and the spread of the results is very small (nearly invisible). The estimates for $\widehat{S^H_{X^k}}$ based on only L samples (blue circles) are also around zero with very small spread. Note that due to the numerical implementation, the sensitivity indices can become negative.

For the estimates based on the Gaussian process (orange squares), the situation is a little different because the Gaussian process fits a function through the data while there is no functional relation between input and output. Hence, the sensitivity index will most likely not be equal to zero. When fitting the Gaussian process, two cases appear, which have the same effect. The length scale and the process variance are either both small or both large. As a result, the predictions based on the Gaussian process (both the mean and the samples) will be inaccurate. This can be seen in the figure by their mean being away from zero and the large spread of their estimates. We included the results from (5.5) and (5.6) despite the bad fit, mainly to show the bias it leads to.

The results for PDD-KDE-MC (red pentagons) also show a large spread and a deviation from zero, hence we adapted the y-axis for clarification, because the largest plot value was a little above 0.2. We hypothesize that the larger spread is caused by the PDD-KDE-MC method fitting a function through the data. This is also the case for Gaussian process methods, although the Gaussian process methods interpolate the data, which the PDD-KDE-MC method does not.

The (F)MST-based sensitivity index is computed on the full data with output predicted by the emulator as in (5.5). However, due to the nature of this method, high values of $\widehat{S^H_{X^k}}$ are measured because the predicted output values are the values of the prediction mean function, which is a continuous function. Hence, these predictions are located on a curve. Therefore, the values of $\widehat{S^H_{X^k}}$ are too large to be visible in this plot for the chosen values of L, except for $L = 10^3$, where no emulated output is used. For reference, we also computed the FMST-based sensitivity index on the full data without emulator, which gave a reasonable result (purple triangle). We did not include a sample-based FMST estimator, because such an estimator would require multiple estimations of β.

We summarize these results as follows: when an emulator (either Gaussian process or PDD) is used to augment the data for sensitivity analysis, positive values of $\widehat{S^H_{X^k}}$ are found because the emulator is designed to fit a functional relation between input and output. The “sample" method does not suffer from this problem. However, this is a very specific test case in which sample-based estimators are preferred over ones which use an augmented data set.

5.3.2 Analytic test case

We consider a small test case in which we can compute the sensitivity index analytically. The idea behind this is to compare the KDE and the MST method in case no emulator is used. We have

$$\begin{bmatrix} X \\ Y \end{bmatrix} \sim N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}\right).$$

Note the similarity of this test case with respect to the one in Section 2.4.1. We took $N = 10^4$ and repeated the experiment $n_r = 10$ times. The results are in Figure 5.2. Except for $\rho = 0.98$, the MST method outperforms the KDE method.

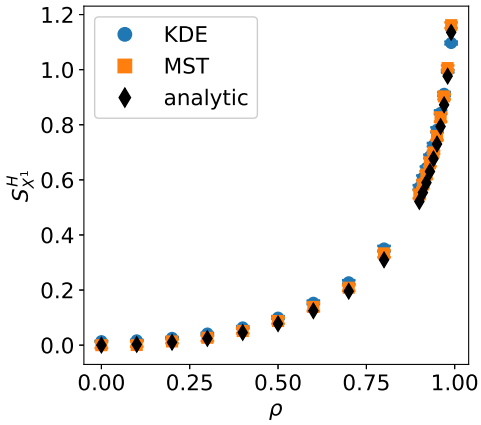


Figure 5.2: Computed values for the sensitivity indices.

Furthermore, the MST method is i) not dependent on parameter choices such as kernel and kernel bandwidth and ii) faster to compute. One also needs to take into account that the rule of thumb to choose the kernel bandwidth we used here is based on the assumption that the data comes from a normal distribution and, therefore, this kernel bandwidth is optimal in this test case. When the underlying distribution is not normal, this heuristic may not be optimal.

5.3.3 Ishigami function

We continue with the Ishigami test case as started in Section 2.5.1 and continued in Section 3.4.4, of which the test function is from Ishigami & Homma [46].

In the numerical experiments, we first compute input samples of sizes $L = \{30, 50, 100, 200\}$ and combine these with KDE to compute (5.4). For the independent data, the samples are obtained with LHS, while for the dependent data, Monte Carlo samples (MCS) are used. Because the samples need to be used in a

kernel density estimation, it is not desirable to use more advanced methods for sample selection such as selecting the centers from a k-means clustering. This because the samples for KDE are assumed to come from the corresponding distribution. The samples generated by more advanced sample selection methods are not per se samples from that distribution. Next, we fit a Gaussian process with Gaussian kernel to these samples, where the length scales are estimated by maximum likelihood estimation.

Now, we can proceed with KDE on (X_+, Y_+), in which we include both the choices $Y_{L^+} = \mu(X_{L^+})$ (5.5) and $Y_{L^+}^{(s)} \sim N(\mu(X_{L^+}), \Sigma(X_{L^+}))$ (5.6). In the first case, we obtain one estimate for $\overline{H}_{X^k,f}^{(L+L_+)}$ for each repetition of the experiment and thereby one value of $\left|\overline{H}_{X^k,f}^{(L+L_+)} - \overline{H}_{X^k,f}^{(N)}\right| \approx |\widehat{S}_{X^k} - S_{X^k}|$, which is used as measure of convergence. Note we use both $L + L_+$ and N in the notation. Although they are equal in value, we distinguish between estimates obtained from output and predictions $(L + L_+)$ and estimates obtained with full output (N). In the second case, we take the number of samples $n_s = 10$. For each sample $Y_{L^+}^{(s)}$ as a prediction of the output, we compute (5.4). This leads to n_s estimates of $\overline{H}_{X^k,f}^{(L+L_+)}$ and the convergence measure $\left|\overline{H}_{X^k,f}^{(L+L_+)} - \overline{H}_{X^k,f}^{(N)}\right|$. Note this value of n_s is relatively low, but each sample requires a new set of kernel density or minimum spanning tree estimates.

The other option is to proceed with the minimum spanning tree method, in which we only use the mean of the Gaussian process as $Y_{L^+} = \mu(X_{L^+})$ (5.5). On the augmented set (X_+, Y_+), we compute the minimum spanning tree length L_γ, which is used in (5.9) for each repetition of the experiment.

Accuracy of the estimated sensitivity indices

The average value and the minimum and maximum obtained value of the sensitivity indices are shown in Figure 5.3. We see that although these sensitivity indices should be equal, there is a discrepancy between the two methods, especially for the second input dimension. From this result, it is not clear immediately which of them is more accurate. Based on Section 5.3.2, we select the MST result to be the more accurate one.

Goodness-of-fit of the Gaussian process

We first show the results for the independent data, followed by the results for the dependent data. We start with determining the goodness-of-fit of the Gaussian process by performing k-fold cross-validation with $k = 10$ and compute the predictivity coefficient Q^2 (which is similar to the coefficient of determination R^2 in

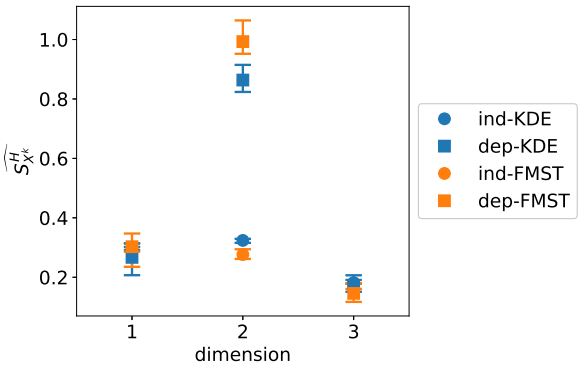
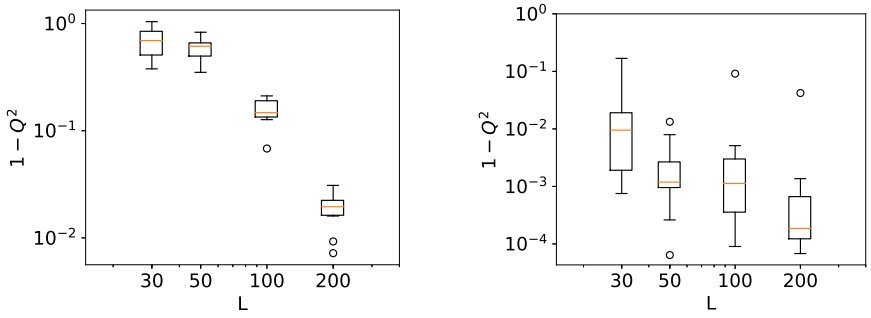


Figure 5.3: Computed values for the sensitivity indices per variable.

regression):

$$Q^2 = 1 - \frac{SOS_{\text{res}}}{SOS_{\text{tot}}} = 1 - \frac{\frac{1}{L}\sum_l (\hat{Y}_l - Y_l)^2}{\frac{1}{L}\sum_l (Y_l - \bar{Y})^2}, \tag{5.10}$$

where SOS_{res} and SOS_{tot} denote the residual and total sum-of-squares, respectively. $\hat{Y}_l$ are the cross-validation predictions for Y_l and $\bar{Y} = \frac{1}{L}\sum_l Y_l$. In Figure 5.4, we show $\frac{SOS_{\text{res}}}{SOS_{\text{tot}}}$ and we see its values are near zero for higher values of L for the independent data. For $L = 30$, this fraction can become larger than 1. In this case, the fit is worse than a constant function. Note that this also means that the Gaussian process has not been fit well. For the dependent data, the results are already very accurate for $L = 30$.



(a) Independent data.

(b) Dependent data.

Figure 5.4: Cross-validation results showing the quality of the Gaussian process.

Convergence of the estimates

The convergence of the estimates as L increases is studied. For reference, the KDE method on the full data set is used for all estimators except for the one based on the MST method. There, the MST method on the full data set is used for reference. This is done because the estimates for the full set differ in value, as seen earlier in Figure 5.3. We start with the case of independent input data. Figure 5.5 shows the convergence of the estimates, where "sample" indicates the KDE is based on only L samples (i.e., without using the Gaussian process surrogate), "GP-mean" is based on (5.5) and "GP-pred" is based on (5.6). Therefore, "GP-mean" uses only the mean of the Gaussian process, while "GP-pred" uses predictions from the distribution. "PDD" denotes the PDD-KDE-MC method, while "MST" denotes the minimum spanning tree method. From left to right, variables 1 to 3 are shown. This will also be the case for all similar figures in this section.

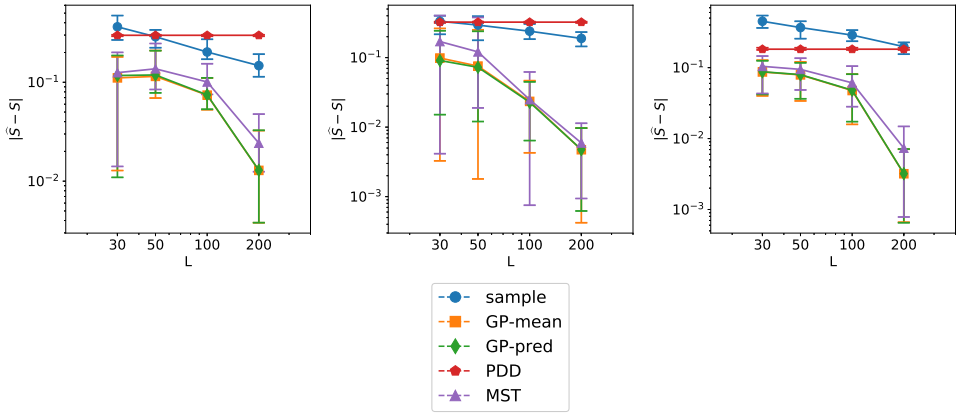


Figure 5.5: Convergence of the estimates for the sensitivity index, independent data.

We see the differences are small for low values of L, while for higher values of L, the estimates based on Gaussian processes are remarkably better. The convergence using the MST method is a little slower than with the other Gaussian process methods. Also, the mean of the sampled estimates matches the estimated mean function within sampling error, as expected. Figure 5.6 shows the average and the minimum and maximum value (determined from the n_r repetitions) of the estimated standard deviation of $\hat{S}$ for the sampled estimates (i.e., the standard deviation of $\hat{S}$ based on the n_s samples of $\hat{S}$). The standard deviation decreases very fast for an increasing number of samples L. The standard deviation of the estimates for $\overline{H}_{X^k,f}^{(L+L_+)}$ is not enough for the 95% confidence interval for $\overline{H}_{X^k,f}^{(L+L_+)}$ to include the reference value. This is due to the bias caused by the use of the surrogate model rather than the exact output function. Although $S_{X^k}^f$ is nonnegative,

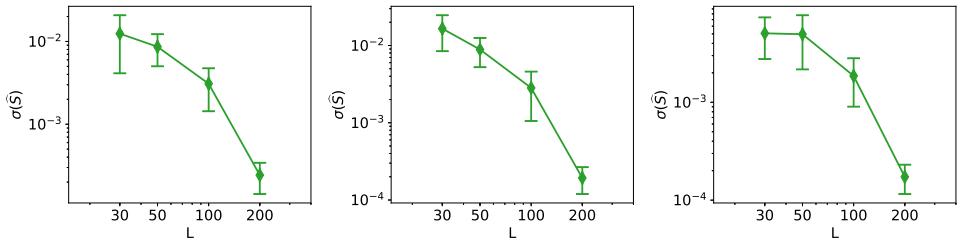


Figure 5.6: Behavior of the standard deviation of the estimates based on predicted output samples for the sensitivity index, independent data.

its estimates $\overline{H}_{X^k,f}^{(L+L_+)}$ can become negative due to the numerical implementation.

The results for dependent data are shown in Figures 5.7 and 5.8. Note that LHS is not an appropriate sampling method in this case because the data is dependent, therefore, Monte Carlo sampling is used instead. For the same reason, PDD-KDE-MC is not suitable here. The results are similar to previous experiments.

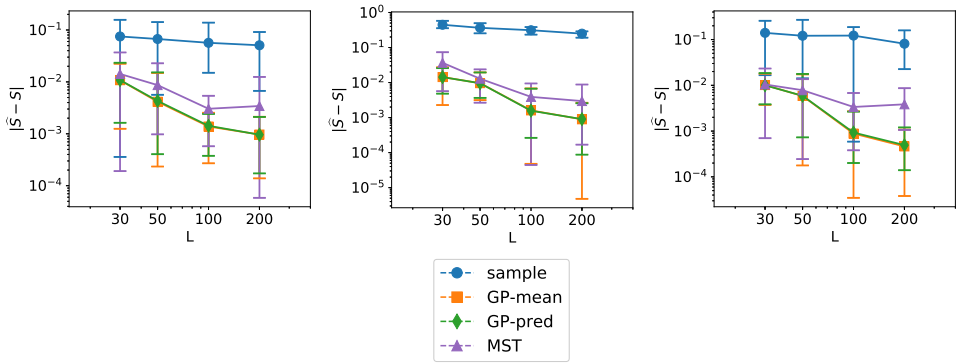


Figure 5.7: Convergence of the estimates for the sensitivity index, dependent data.

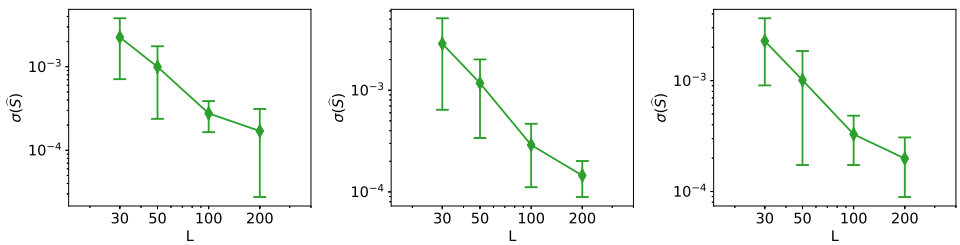


Figure 5.8: Behavior of the standard deviation of the error in the estimates based on predicted output samples for the sensitivity index, dependent input data.

5.3.4 Recommendation

The proposed methods perform significantly better than the PDD-KDE-MC method in the case of a low number of available input-output samples. Although the PDD-KDE-MC method can be performed much faster than the Gaussian process-based methods, its results are less accurate. Since the computational model will have a relatively high cost, the smaller computational cost of PDD-KDE-MC with respect to Gaussian process-based methods is not a real advantage in practice. Furthermore, the MST-based method is in general more accurate and faster than the KDE-based method. Hence, although the convergence for MST may be a little slower than for KDE, the bias of MST is smaller than the bias of KDE. In this light, we propose to use the MST method to compute the divergence-based sensitivity indices.

5.4 Direct sensitivity indices

We note that the sensitivity indices as described by [101] are total sensitivity indices, which include both direct and indirect effects. Direct effects measure the effect of one input variable only, while indirect effects contain the effect of the other variables due to possible dependencies in the input variables. The indirect effect is the difference between the total and the direct effect. To illustrate this, consider an example where $X = (X^1, X^2)$ follows a bivariate normal distribution with means 0, variances 1 and covariance $\rho > 0$ (so that X^1 and X^2 are dependent), while $u(x_1, x_2) = 2x_1$. Then the direct effect of X^2 is zero, while its total effect is positive (because $u(X^1, X^2)$ and X^2 are dependent through X^1). Hence, the indirect effect of X^2 is positive as well. Our goal is now to find a measure for the direct effects, i.e., without the effects of the mutual input dependencies.

Although useful, total sensitivity indices do not tell the complete story. While an input variable may be completely irrelevant for the value of the output, it may have a positive sensitivity index due to a dependency with a relevant input. The relevant input variable would then be called a confounder. An example of this is wave height for a computational model of offshore wind energy: although the waves have nothing to do with the power output, they are linked to each other via the wind speed with which they have a dependency. To get rid of this effect, we need to construct indices which measure the effect of only one input variable, without effects due to dependencies in the input. It is in this case necessary to remove the dependencies from the input.

For variance-based sensitivity indices, a distinction is made between first-order, higher-order and total sensitivity indices [47]. In first-order indices, one only measures the effect of varying one variable alone, where in higher-order indices

multiple variables are varied at the same time. Because the number of second-order sensitivity indices grows as $d(d-1)/2$ with d the number of input variables, and the total number of sensitivity indices is $2^d - 1$, usually not all of them are computed. Instead, one computes the first-order and total sensitivity indices.

In a similar fashion to first-order indices, we define direct sensitivity indices, which measure the effect of varying one variable only. The direct indices then measure the direct effects, while the total indices measure the combination of direct and indirect effects, which also includes effects due to dependencies in the input.

5.4.1 Theory

The starting point of these new indices is the same divergence-based index as before, namely (5.8). We repeat (5.8) here for convenience,

$$S^H_{X^k} = 2 - 2\int\int_{\mathbb{R}^2} \sqrt{p_{X^k}(x)p_Y(y)p_{X^k,Y}(x,y)}\mathrm{d}y\mathrm{d}x.$$

Now, note that

$$Y = u(X^1,\ldots,X^d) = u(X),$$

with $u(\cdot)$ being the model used to obtain the output Y, hence, both $p_Y(y)$ and $p_{X^k,Y}(x,y)$ depend in theory on all input variables X^k. Hence, if we remove the dependencies between the input variables, then these probability distributions change as well. This removal is done by applying a permutation operator Π, which is defined on a data set X in such a way that

$$\Pi(X) = (\Pi(X^1),\ldots,\Pi(X^d)),$$

with

$$CDF(\Pi(X^k)) = CDF(X^k), \quad \Pi(X^i) \perp \Pi(X^j) \text{ for all } i,j, \quad i \neq j.$$

Hence, this operator keeps the marginal distributions the same, but it removes all dependencies ($\perp$ here denotes statistical independence). The implementation of this operator is detailed at the end of this section.

Now, we create a permuted version of our data set X, being $\Pi(X)$. For this data set, we can define the direct sensitivity index by

$$S^H_{D,X^k} = 2 - 2\int\int_{\mathbb{R}^2} \sqrt{p_{\Pi(X^k)}(x)p_Y(y)p_{\Pi(X^k),Y}(x,y)}\mathrm{d}y\mathrm{d}x.$$

The output $Y = u(X)$ can be replaced by

$$\widetilde{Y} = \tilde{u}(\Pi(X)),$$

in which $\tilde{u}(\cdot)$ denotes the Gaussian process $\widetilde{G}_{\{X_L,Y_L\}}(\cdot)$ constructed earlier. This leads to the estimator

$$\overline{S}^H_{D,X^k} = 2 - 2\iint_{\mathbb{R}^2} \sqrt{p_{\Pi(X^k)}(x)p_{\tilde{Y}}(y)p_{\Pi(X^k),\tilde{Y}}(x,y)}\mathrm{d}y\mathrm{d}x.$$

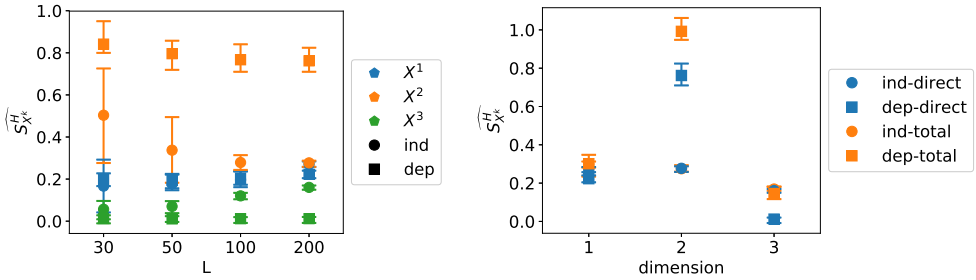
The problem now is how to define $\Pi(X)$. A naive implementation could be one in which for each variable, a random permutation of the values is performed. This is fast, but does not guarantee independence of the input variables after transformation. Also, the indexing of the permutations leads to a Latin hypercube design (LHD): each value from 1 to N (for N data points in the data set) is used only once. However, this does not guarantee all dependencies are removed. In Latin hypercube sampling (LHS), a comparable problem exists as equally probable subspaces can end up with a different number of sampling points. This is solved by orthogonal sampling [113] or by using a maximin criterion [114].

Inspired by this, we would like to generate an LHD of size N in d dimensions with the maximin criterion which puts the samples at the middle of each interval. This LHD is easily transformed to an indexing, which can be applied to the original data X to obtain $\Pi(X)$. However, obtaining such an LHD is computationally very expensive because it contains an optimization step and is therefore not feasible for the problem sizes we are looking at.

An alternative to Latin hypercube sampling is quasi-Monte Carlo sampling, which generates data points from low-discrepancy sequences such as Halton's [115, Chapter 3] and Sobol's [116]. In this way, we achieve the goal that the proportion of data points in a sequence falling into a subspace is nearly proportional to the probability measure of this subspace (the difference between them is the discrepancy). Hence, we achieve an approximately uniform distribution of data points over the unit hypercube, which means the dimensions are independent of each other. Furthermore, all values generated for a variable are unique, which means they can easily be transformed to the discrete hypercube $\{1,\ldots,N\}^d$. The transformed values can be used as an indexing for X to obtain $\Pi(X)$. Because the data points generated by the sequence are uniform over the unit hypercube, they lead to an independent data set when their transformed values are used as indexing. We use the Sobol sequences as described by [116].

5.4.2 Ishigami

We apply this method to the Ishigami test case described in Sections 2.5.1, 3.4.4 and 5.3.3, in which both dependent and independent data is used. The results are shown in Figure 5.9. Figure 5.9a shows that for increasing L, the spread of the estimates decreases in general. Also, we see a difference between the direct indices for the independent and dependent data. This is expected and it is due to



(a) Convergence for increasing L.

(b) Comparison with the total indices.

Figure 5.9: Direct indices for the Ishigami test case.

the effect of the marginal distributions, which are uniform in the former case and normal in the latter.

We can compare these values to the total indices which were shown in Figure 5.3, as we do in Figure 5.9b. The orange markers in Figure 5.9b match the ones in Figure 5.3. First, we see the total indices are larger than the direct indices, except for the independent data in variable 2. However, the difference is small and can be caused by the sampling, as the data sets are not completely the same due to the transformation by the low-discrepancy sequence. Also, for the other variables in the independent set, we expect the values to be approximately equal, which is indeed the case, as some of the markers overlap. For the dependent set, we see the indices are larger for the total indices than for the direct indices, which is caused by the dependencies. Note it is possible to have direct indices larger than total indices, as dependencies can also attenuate the effects of different input variables.

In short, the results of the direct indices are as expected and consistent with how such indices should behave. Another application can be found in Section 7.3.7.

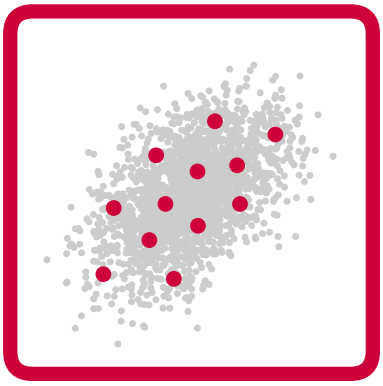
5.5 Conclusion

We proposed to use Gaussian processes in order to improve the estimates of divergence-based sensitivity indices. This is advantageous in cases where the number of available input-output samples is small, for example if the computational cost of each model evaluation needed to compute the output is high. Two estimators based on kernel density estimation were investigated, in which one uses the prediction mean of the Gaussian process and the other uses the complete multivariate normal distribution given by the Gaussian process from which multiple samples were obtained. The first of these is also computed by the minimum spanning tree method from Chapter 2. In terms of accuracy, the minimum spanning tree-

based method performs better than the kernel density estimate-based method and much better than the reference method based on polynomial dimensional decomposition.

However, we have shown that the bias due to the use of the Gaussian process instead of the exact output function, can be larger than the width of the estimated confidence intervals and can thereby provide false confidence in the results. Therefore, we do not advise to use the sample-based method. Instead, we advise to use the prediction mean of a well-fit Gaussian process to improve the estimates of divergence-based sensitivity indices computed by minimum spanning trees.

Furthermore, we developed a new type of sensitivity indices to distinguish between direct and indirect effects. In this way, some issues regarding causality can be solved and more insight in the results can be obtained.



6 Adaptive sampling

This chapter introduces a way to add extra samples to an existing sampling design for surrogate modeling irrespective of the way these initial samples are obtained. The addition of samples makes use of information hidden in the available input-output samples by using Gaussian process length scales.

6.1 Introduction

Chapter 3 describes sample selection methods for dependent input variables. In practice, an initial sampling design may not deliver sufficiently accurate results, instead requiring more evaluation data of the computational model. Additional sample points must be obtained at wisely chosen locations in the input space to benefit most.

The original design is often referred to as a one-stage (sampling) design, because it is chosen at once. This in contrast to an online (sampling) design, which starts with an initial design and adds sample points one-by-one or in small batches. When the new samples are based on information from the earlier ones, one can call these methods adaptive sampling methods.

The idea of using adaptive sampling methods is not new. Such methods are useful because information on the output can be incorporated to refine the experimental design. Several efforts are directed to adaptive sampling for design optimization [117–119]. Some methods used in these efforts make use of (dynamic) radial basis functions [120, 121] and trust-region methods [122]. Recent applications can be found in [123] for computational fluid dynamics and in [124] for production optimization in oil reservoirs. However, most efforts are directed towards design optimization rather than accurate prediction over the complete input domain. This means that samples are added at locations that help in finding an optimal design rather than decreasing prediction uncertainty over the complete domain. We mention the review by Queipo et al. [125] on surrogate-based

optimization, and the more recent one by Liu et al. [126], for global emulation for complex simulation-based engineering design.

In contrast to other methods, we use an adaptation of k-means clustering as the main tool to construct additional samples. We introduce two methods: one based solely on the locations of the earlier input samples and one based on both the locations and the output values of the earlier input samples. The difference is in the norm used in the k-means clustering. The first method uses the Euclidean distance, while the second one uses the Mahalanobis distance based on the length scales of a Gaussian process, fit on the available input-output samples. Note this distance agrees to the distance used earlier (4.2) in our description of Gaussian processes.

One may ask why we base this method on an adaptation of the k-means clustering. The motivation is threefold: first, k-means clustering proved itself as a useful sample selection method (Chapter 3). Second, because choosing k-means clustering as sample selection method is a good heuristic to decrease the prediction variance of a Gaussian process because of the link between k-means clustering and prediction variance, as detailed in Section 4.5.3. The third reason is the determination of the additional samples at once, thereby giving the option to parallelize the evaluation of the computational model.

Section 6.2 explains the methods and the UQ framework they fit in, while Section 6.3 shows some test results. Section 6.4 concludes.

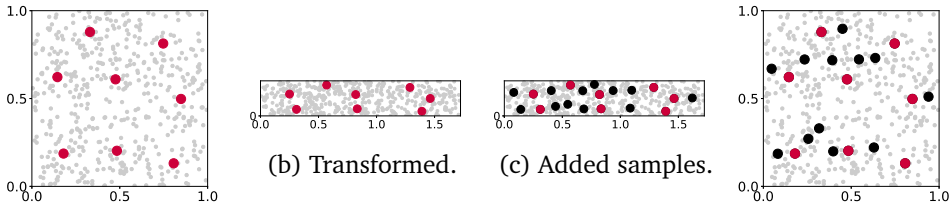
6.2 Methods

We first give the two-stage designs in Section 6.2.1, after which we explain the adaptive k-means clustering in Section 6.2.2.

6.2.1 Two-stage design

The two-stage design consists of the following steps:

1. Obtain an initial design of size L_0, $Z = \{\mathbf{z}_1, \ldots, \mathbf{z}_{L_0}\}$.
2. Evaluate the output for the initial design, $Y = \{u(\mathbf{z}_1), \ldots, u(\mathbf{z}_{L_0})\}$.
3. Fit a Gaussian process to the output (optimize $\mathbf{l}$ and compute $\widehat{\sigma^2}$).
4. Construct the remainder of the design $Z^* = \{\mathbf{z}_{L_0+1}, \ldots, \mathbf{z}_L\}$ with the original design Z fixed.
5. Evaluate the output Y^* for Z^*.
6. Construct the final Gaussian process (if desired).



(a) Initial samples. (b) Transformed. (c) Added samples. (d) Final samples.

Figure 6.1: Illustration of the sampling process. We start with the initial samples (red dots) constructed on the gray data and transform them using the length scales $\mathbf{l}$. Then, we construct the remaining samples and transform them back.

For the non-adaptive design, i.e., in which no information on the output values of the initial sample is used, Step 3 is left out. Note Steps 1 and 2 are already done when the adaptive part starts. Furthermore, we assume the data is properly scaled. Step 4 is given in Section 6.2.2.

The steps are illustrated on the basis of Figure 6.1. In this example, we first perform k-means clustering with $L_0 = 8$ clusters on the input data to give us the first L_0 samples (Figure 6.1a, Step 1). The complex model is evaluated on these samples (Step 2). Then, we fit a Gaussian process through the samples and their output (Step 3). The length scales obtained in this step are used in the Mahalanobis distance as chosen in (4.6). To show the effect of the length scales, we have scaled the domain with these length scales, such that regular Euclidean distance on this domain equals the Mahalanobis distance on the original domain (Figure 6.1b). The Gaussian process fitting is followed by using the adapted k-means clustering for $J = 20$ (Figure 6.1c, Step 4). The obtained samples in the original space are shown in Figure 6.1d. In this example, we leave out the second output evaluation (Step 5) and the construction of the final Gaussian process (Step 6). We emphasize that applying k-means with the used Mahalanobis distance is identical to applying k-means with Euclidean distance after a suitable scaling. The "complex model" used in constructing this figure is the output function $f(x_1, x_2) = \sin(4x_1 + 0.5) + x_2$.

In Figure 6.2, we show the result of adding samples when the length scales are not used. In this case, the samples are more spread throughout the sample and there is no visible anisotropy.

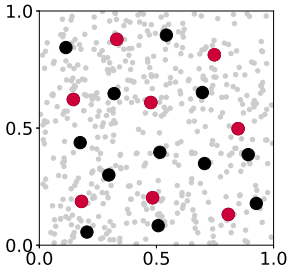


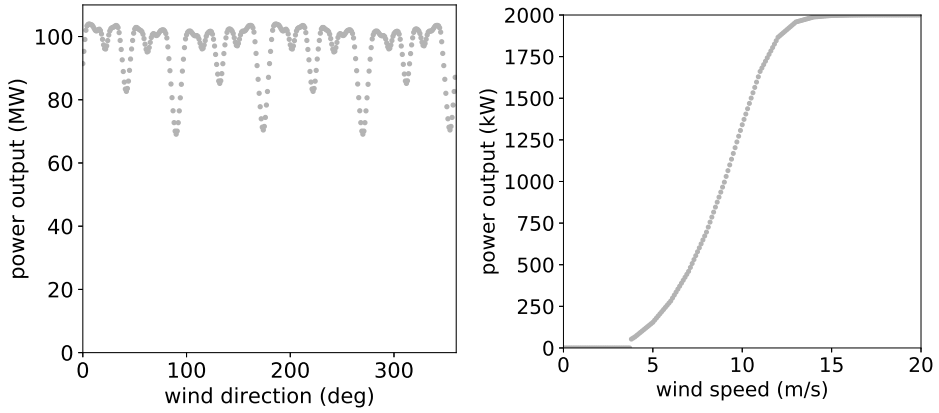
Figure 6.2: Adding samples without scaling.

6.2.2 Adaptive k-means

The cluster centers of k-means clustering are proposed as a sampling design in Chapter 3, because this design method is able to handle dependent input variables, regardless of the type of dependence. It is also chosen for the second stage of the two-stage design because it minimizes the sum-of-squares, and, as shown in Section 4.5, this tightens the bound on the prediction variance. One disadvantage of k-means is the property that it can end up in local minima, but this is usually mitigated by restarting the algorithm several times with different initializations. For using it in the second stage, some adaptations are necessary, because the initial design poses a problem for the k-means in the second stage (Step 4).

In each step of the k-means algorithm, all data points are assigned to the nearest cluster (center), after which the cluster centers are recomputed. Hence, it is very likely for the cluster centers at the end of the algorithm to differ from the ones in the beginning. If the initial samples are included as initial cluster centers, then it is most likely that these are not cluster centers anymore at the end of the algorithm. This would be a waste of resources as the corresponding samples have already been evaluated with the computational model $u(\cdot)$.

We therefore adapt the k-means algorithm such that these initial samples are not allowed to change. We start the algorithm by the k-means++ initialization where we already included the initial samples. In each step of the algorithm, all data points are still assigned to the nearest cluster (center), but only the newly added cluster centers are allowed to be recomputed. Thus, the minimization of the sum-of-squares in the k-means algorithm is carried out over the coordinates of the new cluster centers only. It can be proven that the algorithm still converges, albeit maybe to a(nother) local minimum.



(a) Wind direction.

(b) Wind speed.

Figure 6.3: Behavior of the power output with respect to wind direction and wind speed.

6.3 Results

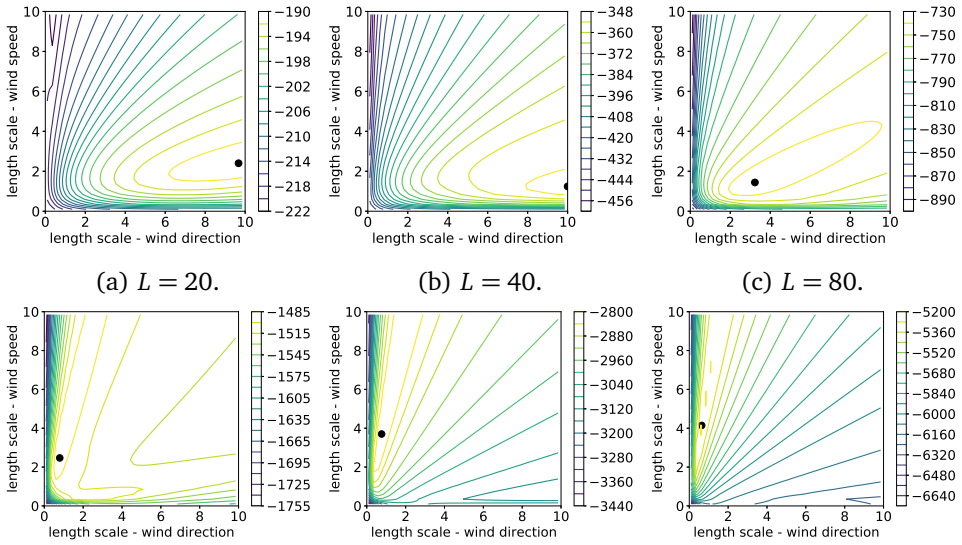
We first show an example where this method does not work due to a problem inherent to sampling. This is meant as a warning to the user to take care that the number of samples in the initial design is high enough to construct a reliable Gaussian process on. Then, we show the method on the Ishigami example used before.

6.3.1 Meteomast IJmuiden

The data for this test case is a subset of size $N = 10^4$ with $d = 2$ obtained from the data used in Chapter 7. In short, it contains data on wind direction and wind speed measured at a certain height ($h = 58$ m). The output is given by a simulation which computes the power output of a wind farm under these circumstances, as detailed in Section 7.2.3. In Figure 6.3a the behavior of the power output with given wind direction for a complete wind farm is shown, while Figure 6.3b shows the behavior of the power output given wind speed for a single wind turbine. One can see that the length scale with respect to the size of the domain will be shorter for the wind direction than for the wind speed. Also, the wind speed shows a discontinuity at the cut-in wind speed, while the output function is smooth in the wind direction. The problem here is, that with a small value of L the short length scale for wind direction will not be picked up in the length scale optimization. In that case, the behavior of the output with respect to wind direction is considered noise rather than a meaningful signal.

Figure 6.4 shows this effect. The data has been normalized to zero mean and

unit variance, after which the log-likelihood of the length scales is determined. The optimum is given as a black dot. For low values of L, the small length scale for the wind direction has not yet been recognized, while this is the case for higher values of L. Therefore, the adaptive sampling does not work well for small L because the estimated length scales are not stable yet due to a lack of information. In practice, to assess this, one has to estimate the length scales on subsets of the samples to find out whether the length scales are already stable. If this is not the case, the number of samples is insufficient for accurate Gaussian process estimation.



(a) $L = 20$. (b) $L = 40$. (c) $L = 80$.

(d) $L = 160$. (e) $L = 320$. (f) $L = 640$.

Figure 6.4: Illustration of the change in the log-likelihood, the black dot represents the maximum.

6.3.2 Ishigami

We continue with the example from Section 5.4.2. Because the previous example showed that length scales may change substantially for varying L, we show the estimates for $\mathbf{1}$ for all $n_r = 10$ repetitions in Figures 6.5 and 6.6. Here, we did not use the adaptivity yet, we only varied L. These simulations will be used as a reference for the results obtained with adaptive sampling. The spread in the estimates is large for small values of L and decreases for increasing L. This motivates us to continue and we compute the prediction coefficient Q^2 according to (5.10) based on all reference data. The results are shown in Figure 6.7, where we plot

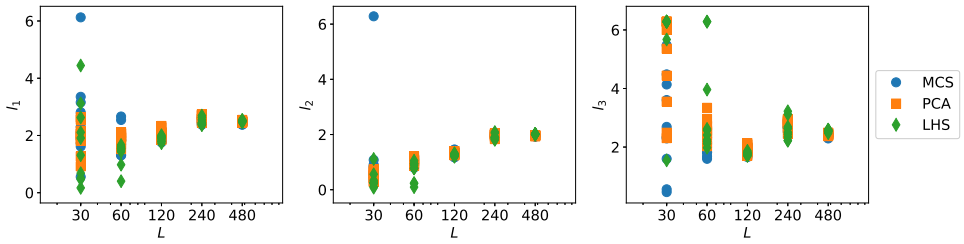


Figure 6.5: Estimates for the different l_i for independent data. From left to right, the variables 1-3 are shown.

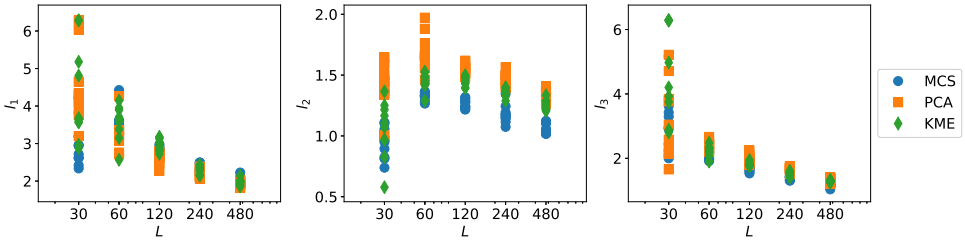
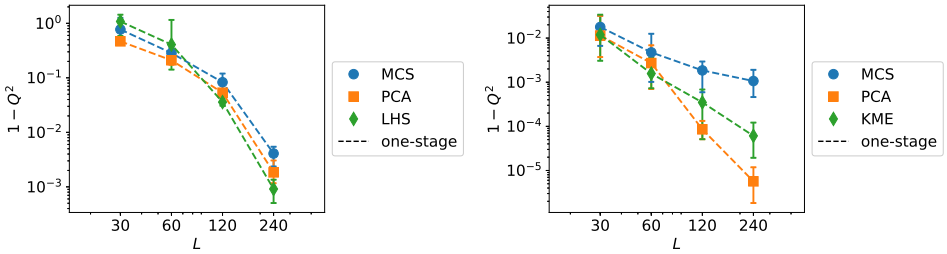


Figure 6.6: Estimates for the different l_i for dependent data. From left to right, the variables 1-3 are shown.



(a) Independent data. (b) Dependent data.

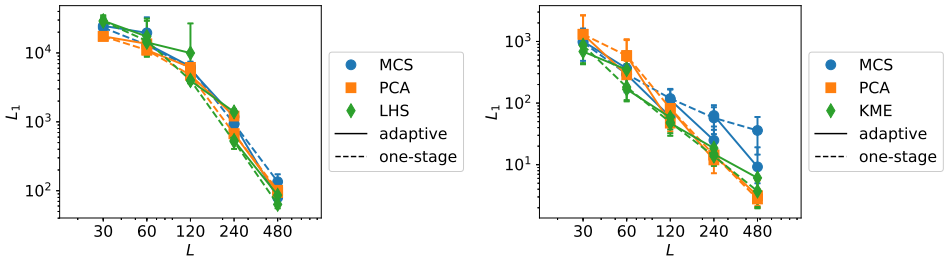
Figure 6.7: Validation results showing the quality of the Gaussian process for the one-stage design.

$1-Q^2$ instead, which should be near-zero for accurate estimates. In this case, we did not use cross-validation, but computed the real values Y_l for all data points and included all data points in this coefficient. In a real-life situation, this is not possible and one should resort to cross-validation.

Values below 0.01 indicate very good correspondence, so $L = 240$ is here sufficient for the independent data and $L = 120$ for the dependent data.

We now continue to the results for both one-stage and adaptive sampling,

in which $L_0 \in \{30, 60, 120, 240\}$ and choose $L = 2L_0$. The independent data has its initial stage computed by LHS, while the dependent data uses KME. Note the Gaussian process may not be sufficiently accurate according to Q^2, but we investigate here its quality by the L_1-norm between the real and the emulated output. For the second Gaussian process, the original length scales are reused. The results are in Figure 6.8. We do not see a clear difference between one-stage

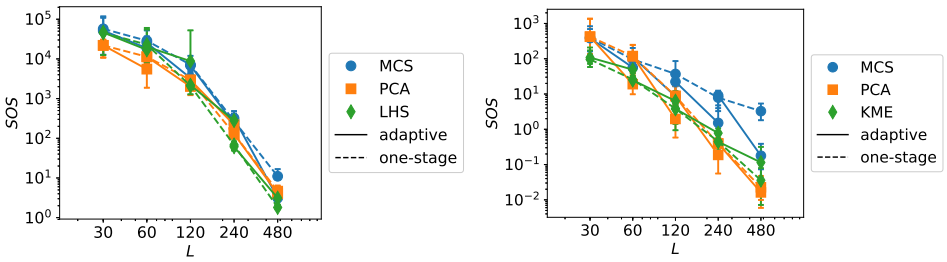


(a) Independent data. (b) Dependent data.

Figure 6.8: Convergence of both one-stage and adaptive sampling.

and adaptive sampling in terms of convergence. Although this means there is no direct benefit from the two-stage method, this also shows that in case the initial design is augmented with extra samples, the resulting design is not worse than a one-stage design. The number of available samples can therefore be used frugally, i.e., not use all samples immediately, but save some in case they turn out to be necessary. This approach may be used in situations where computational power needs to be divided over several projects. Furthermore, when the length scales are less similar to each other, the two-stage method may have greater benefits.

Finally, we show the sum of the prediction variances (being a sum-of-squares) for the same simulation experiments in Figure 6.9. Here, we see improvements for MCS, especially when L is large. This can be seen by the fixed lines being



(a) Independent data. (b) Dependent data.

Figure 6.9: Sum of the prediction variance for both one-stage and adaptive sampling.

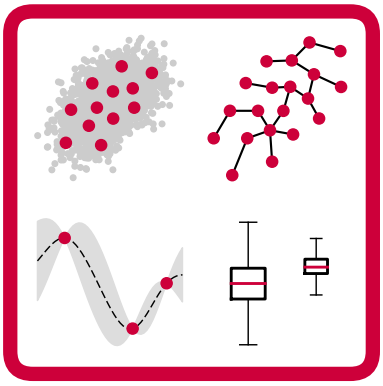
below the dashed lines. This indicates that although the prediction quality of the Gaussian process is relatively unaffected by the choice of sampling method, the prediction variance can decrease by a substantial quantity in this adaptive sampling. Note this was one of the reasons to use k-means clustering for the second stage.

6.4 Conclusion

We have introduced a way to add extra samples to an existing sampling design. This method is based on an adaptation of the k-means clustering method which incorporates the length scales of an emulator constructed as a Gaussian process on the existing design. The idea is twofold: on one hand, these length scales contain information on the output, which is incorporated in this way, while on the other hand, the choice to adapt the k-means clustering results in a well-spread design (i.e., a low sum-of-squares between data points and samples) which decreases the prediction variance of the Gaussian process.

In practice, there are a few drawbacks to this method. In the results, we show that these length scales can change for an increasing number of samples and even change behavior completely if the original number of samples is very small. Furthermore, it may be that the spread of the design points is not crucial to obtain a well-predicting Gaussian process when enough samples are available, although a better spread will lead to a lower prediction variance.

The results also show that this method does not perform better than corresponding one-stage design methods, although the prediction variance decreases by a substantial amount in case the original design came from Monte Carlo sampling. In this situation, the method can be beneficial in terms of uncertainty reduction.



7 Case study I

In this chapter, we consider a case study which encompasses the topics studied in Chapters 2, 3 and 5. This case study considers the power output and efficiency of a hypothetical wind farm with the layout of the offshore wind farm Horns Rev I located at Meteomast IJmuiden.

7.1 Introduction

Uncertainty in wind and wave conditions during the turbine life is one of the reasons for the high cost of offshore wind farms. Although these uncertainties cannot be eliminated, it is crucial to know their effects on both the loads of the turbines and the power output. Since the power output is roughly proportional to the third power of the wind speed, small differences in wind speed statistics can lead to substantial differences in the mean and variance of the power output, potentially making the difference between a profitable or unprofitable wind farm. Numerical simulation of a wind farm under different wind and sea conditions can be a valuable tool for decision making. However, because of the complexity of the system which consists of weather conditions and turbines, it is not feasible to perform simulations for all possible or observed input conditions.

It is therefore important to select in a careful way the input conditions for which the wind farm simulation model is going to be evaluated (we refer to these selected input conditions as samples hereafter). Possible dependencies in the input conditions need to be taken into account for this selection. Predictions of the output for input conditions not included in the samples can be made with the use of an emulator. These predictions are used in performing the sensitivity analysis.

The aim of this chapter is to combine the results from previous chapters into the framework as given in Figure 1.3, in which the analysis of the input data

Partially based on A. Eggels and D. Crommelin, "Uncertainty Quantification with dependent inputs: wind and waves," in *Proceedings of ECCM 6 and ECFD 7*, 2018.

and the model output is combined. We illustrate this framework on a wind farm simulation based on the layout of Horns Rev I, with wind and wave data obtained at the Meteomast IJmuiden serving as input. The simulation model that we use is not very advanced, however, our goal is to demonstrate the proposed framework rather than to obtain highly realistic results; the limitations of the model are less important for the present study.

Section 7.2 describes the weather data, the wind farm and the simulation to obtain the power output and efficiency of the wind farm under the given weather conditions. Section 7.3 shows the results, while 7.3.1 describes the methods in the framework. The conclusion follows in Section 7.4.

7.2 Setup

We briefly describe the data in Section 7.2.1, the wind farm that we simulate in Section 7.2.2 and the simulator (computational model) itself in Section 7.2.3.

7.2.1 Meteomast IJmuiden

Meteomast IJmuiden (MMIJ) is a meteorological mast in the North Sea, located approximately 75 km west of the coast of IJmuiden (the Netherlands) [15]. We use data from a measurement campaign that has run from November 2nd, 2011 up to and including March 11th, 2016. Atmospheric conditions (e.g., wind speed and direction, air pressure, air temperature) have been measured at several heights in and above the mast. The data has been post-processed and is publicly available online, see [15]. The list of measurement variables used in our analysis can be found in Appendix B.

7.2.2 Horns Rev I

Horns Rev I is an offshore wind farm in the Danish North Sea consisting of 80 Vestas V80 2MW turbines [127] in an oblique quadrilateral layout with a spacing of 560 m [128], as shown in Figure 7.1. The rotor diameter is 80 m and the hub height is 70 m. It is very well known in the wind energy community [16–19], especially for its wind turbine wakes affecting nearby turbines. We will use its layout for our computational model, described in the next section.

7.2.3 Wind farm simulation

We use the observation data from MMIJ as input for a computational model of Horns Rev I. The output of this model consists of power output and efficiency of the wind farm, given the wind speed and wind direction at various heights as

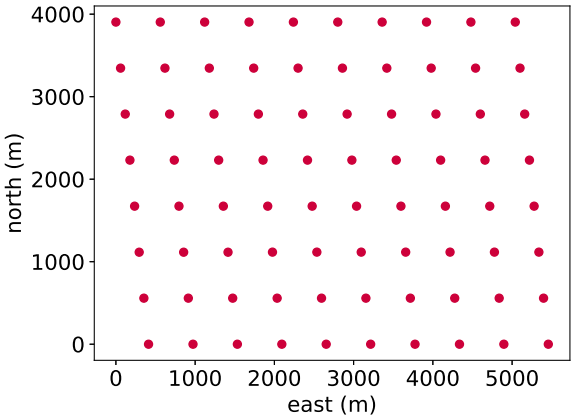


Figure 7.1: Layout of the wind farm used.

input. The computer model is described in [129]. It makes use of a wake model from Bastankhah and Porté-Agel [130]. This model is derived from mass and momentum conservation in case viscous and pressure terms are neglected. The wake expands linearly and the velocity deficit is assumed to be Gaussian. The normalized velocity deficit is then given as

$$\frac{\Delta U}{U_0}(x,y,z) = \left(1-\sqrt{1-\frac{C_T}{8w(x)^2}}\right)\exp\left(\frac{-1}{2w(x)^2}\left(\left(\frac{z-z_h}{d_0}\right)^2+\left(\frac{y}{d_0}\right)^2\right)\right),$$

in which x, y and z represent the streamwise, spanwise, and vertical distance to the rotor, respectively. C_T is the thrust coefficient of the turbine, z_h hub height and d_0 the diameter of the wind turbine. Furthermore, $w(x)$ is given by

$$w(x) = \frac{\kappa^* x}{d_0} + \varepsilon,$$

in which κ^* is the wake growth rate and ε a constant. This constant can be determined by investigating the case where $x \downarrow 0$. Near the rotor, a uniform shape of the wake is preferred over a Gaussian one. Therefore, to determine ε, the Frandsen wake model [131] is used, which is derived from the same governing equations. From here, it is found $\varepsilon = \sqrt{\beta}/4$, in which

$$\beta = \frac{1}{2}\left(\frac{1+\sqrt{1-C_T}}{\sqrt{1-C_T}}\right).$$

This constant can be interpreted as the ratio between the wake cross-sectional area at $x = 0$ and the rotor diameter. From simulations, [130] showed $\varepsilon = 0.2\sqrt{\beta}$

gives a better estimation. The growth rate κ^* is estimated by [132] as

$$\kappa^* = 0.3837I + 0.003678,$$

in which I is the local turbulence intensity given by [133]

$$I = \sqrt{I_0^2 + I_+^2}.$$

Here we chose $I_0 = 0.075$ and I_+ is the turbulence added by the wake, determined from the turbulence model developed by Crespo & Hernandez [134], given by

$$I_+ = 0.73a^{0.8125} I_0^{0.0325} \left(\frac{x}{d_0}\right)^{-0.32}, \quad a = \frac{1}{2}\left(1 - \sqrt{1 - C_T}\right).$$

The thrust coefficient is determined from the curve given by [127], which is valid for the Vestas V80 turbine.

The effects of multiple wakes are combined as in [132] by

$$U_i = U_0 - \sum_k (U_k - U_{ki}),$$

in which $0 < k < i$, U_i and U_k denote the wind speed at turbines i and k, while U_{ki} is the wake velocity of turbine k at turbine i.

Furthermore, since the incoming wind speed can vary over the rotor-swept area due to wake effects, an equivalent incoming wind speed is necessary. This is especially important when the wind speed is between cut-in and rated wind speed. Because thrust is proportional to the square of the velocity and power output is proportional to the cube of the velocity, the equivalent incoming wind speed is computed as in [135] by

$$U_{eq,T} = \sqrt{\frac{\sum_j U_j^2 dA_j}{A}},$$
$$U_{eq,P} = \sqrt[3]{\frac{\sum_j U_j^3 dA_j}{A}}.$$

Herein, U_j denotes the wind speed at segment j, A_j the area of segment j and A denotes the total area swept by the rotor blades. We chose the segments in the polar coordinate system.

We point out that the computational model is not very advanced and has several limitations. However, it is useful in the present context as our objective is to

test the coupling of several methods for uncertainty quantification and sensitivity analysis, rather than to obtain output for operational purposes. We mention two limitations here. First, the incoming wind speed in the model from [129] is constant over the height of the turbine. We made an adaptation such that the wind speed is interpolated linearly between measurements at different heights. Second, the turbulence intensity is fixed at 0.075, mainly to be in the regime for which the turbulence model [134] was proposed.

7.3 Results

Our objective in this test case is to demonstrate how the previously developed methods can be combined. These methods are all aimed at situations where the input distribution is unknown and where a data set of the (multivariate, dependent) inputs is available instead. The data set is assumed to be large (i.e., a large number of data points). Furthermore, the computational model is assumed to be expensive so that only a small number of model evaluations (or simulations) can be performed. Focal point is the presence of dependencies in the data.

7.3.1 Combination of methods

The methods are combined in the following manner, see also Figure 1.3. First, the input data is analyzed for dependencies, as described in Chapter 2. Here, we already know dependencies exist in this data set, but we want to find them and determine their strength. Due to the dependencies, input samples must be selected as explained in Chapter 3, for which we included the methods MCC, PCA and KME. These samples serve as the input conditions for the computational model. Once the input samples have been selected, the wind farm simulations as described in Section 7.2.3 can be performed. The resulting simulation output serves as the basis for an emulator. The goal of this emulator is to construct an approximation of the output for the input conditions that were not part of the samples (thus, the computational model was not evaluated for these inputs). The output of this emulator is used to perform sensitivity analysis.

We first describe the data set of input conditions in more detail in Section 7.3.2. Then the dependency analysis is performed in Section 7.3.3 before continuing with the evaluations of the computational model. We present the results from the sample selection (Section 7.3.4) and the model evaluations in Section 7.3.5. In Section 7.3.6, we discuss the construction of the Gaussian process emulator. The output from the emulator is further used in Section 7.3.7 for sensitivity analysis.

7.3.2 Data set

We construct three data sets from the given MMIJ data as given in Appendix B. The first data set contains only mast measurements (146686 instances in total) and includes also quantities (e.g., air temperature) that are not used as input for the computational model. The second data set is a subset of the first, restricted to only the input variables required for the computational model. The third data set includes wave variables, but is smaller due to the reduced recording frequency (19353 instances).

7.3.3 Dependency analysis

We quantify the dependency strength between observation variables from data set 3 as $D_\alpha(g, \tilde{g})$ (2.3) by (2.5) and (2.6). The estimate for β is computed as 0.6536 from 10^2 replicates. The results are shown in a heatmap in Figure 7.2, where we have grouped the variables. Furthermore, we have applied a discreteness correction for the wave period variables in the rank-transform which accounts for the fact that they have discrete values due to measurement accuracy. In this correction, instead of placing the data points with the same discrete value at the same location, they are spread out in order to make the marginals more uniform.

As expected, strong dependencies exist between similar variables at different heights, such as wind speed, direction, air density, pressure and temperature. For wind speed and wind direction, these are the lighter triangles in the top-left corner of the figure. The two included turbulence intensities (TIs) are strongly dependent as well. Furthermore, strong dependencies can be seen to exist between air density, air pressure and temperature (lower-right near center). Two other groups of strongly dependent variables are the wave period and height variables (lower-right corner). Dependencies between them exist too. The last identifiable group of dependencies is between wind speeds and wave heights. Some dependencies are seen to exist between the air variables and the wave height, although we have no clear physical explanation for these.

7.3.4 Sample selection

Because of the dependencies in the input data, we select samples using the methods described in Chapter 3. For each of the three data sets, we construct three sets of samples with a varying number (denoted L) of samples: $10d$, $20d$ and $40d$, in which d is the dimension (number of variables) included in the data set. The wind farm simulation ran for all the constructed samples.

We illustrate one point of interest, namely the presence of periodic variables (e.g., wind directions). We give a small example with wind direction and wind

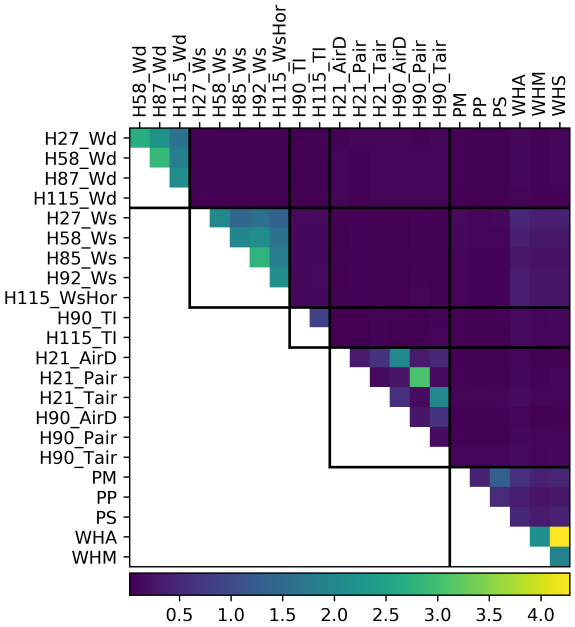


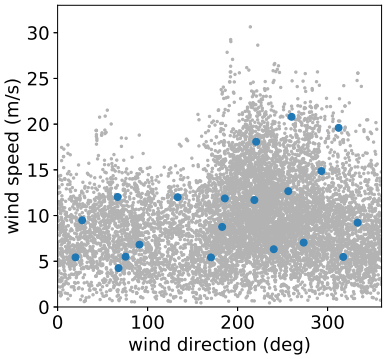
Figure 7.2: Computed values for the quantifier of dependence for data set 3. Lighter squares indicate higher dependence. Because of symmetry, only the upper triangle is shown. The labels are shorthand notations for the variables given in Appendix B.

speed at 58 m height in Figure 7.3. Data points at 1° and 359° are far apart if the periodicity is ignored. However, when multiple variables are periodic, difficulties arise in the computation of distances if the periodicities are taken into account (since the topology is no longer equivalent to $\mathbb{R}^d$). Therefore, we ignore this aspect of directional variables.

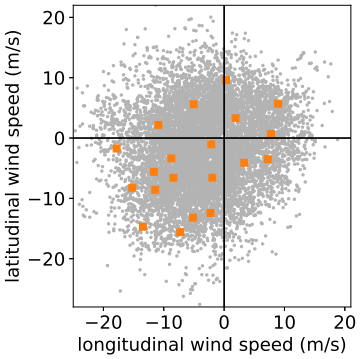
7.3.5 Simulation of the wind farm

The two simulation outputs of interest are the power output and efficiency (the latter being defined as power output relative to maximum power output in case of no wakes). For each sample, the computational model returns one value for the power output and one for the efficiency. For each set of samples, the weighted mean is computed for both output variables. The weights are computed as the fraction of data points associated to each sample.

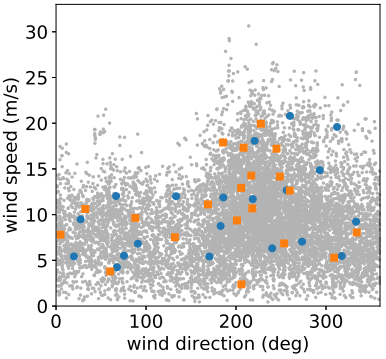
The results for the power output are presented in Figure 7.4. For reference purposes, we have also computed the power output for all data points. In real-life situations, this is not possible. Note the computed reference value is different for data set 1 than for data set 3. This is due to the different measurement peri-



(a) PCA-based clustering ignoring the nature of the data.



(b) PCA-based clustering with the periodicity taken into account.



(c) Comparison of the clusters from (a) and (b). Colors are copied from the corresponding figure.

Figure 7.3: The effect of periodic variables. In (a), the periodicity of the wind direction is ignored, while it is taken into account in (b) by converting the polar coordinates in (a) to a Cartesian grid. They are combined in (c).

ods, because data set 3 only includes data for which all input variables have been recorded. From these results, no method is preferred over another. Figure 7.5 shows the results for the efficiency. Again, no method is preferred over another. This may seem in contrast with the earlier results, where the MCC method performed worse than PCA and KME. It may be explained by the nature of the data, which contains a bulk of regular wind conditions and only a small portion of extreme wind conditions. Although this small portion may be underrepresented in the MCC sampling, due to its small weight, it has no large effect on the output.

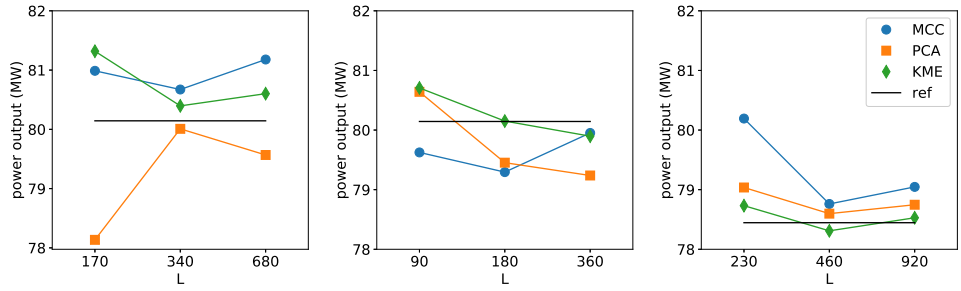


Figure 7.4: Results for the power output based on weighted samples. From left to right, data set 1 to 3 are presented.

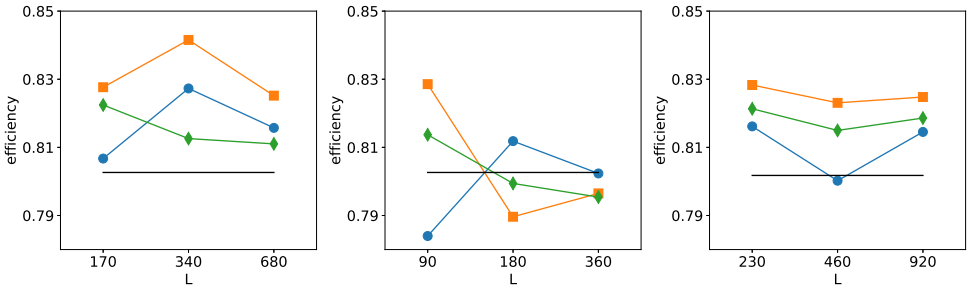


Figure 7.5: Results for the efficiency based on weighted samples. From left to right, data set 1 to 3 are presented. The legend can be found in Figure 7.4.

7.3.6 Emulation of the wind farm simulation

Our implementation of Gaussian processes uses the Matérn kernel with $\nu = 3/2$. We use ordinary kriging, in which the length scales are optimized within bounds from the log-likelihood (as described in Section 4.4.2) with several restarts to avoid local maxima.

The emulator is constructed for the results obtained in the previous paragraph. Altogether, there are 27 different sets of samples, and hence, 27 different emulators for each output. With each emulator, we compute predictions (means and variances) of the power output and efficiency for all data points in the data set associated with the emulator.

Because of the Gaussian properties of each emulator, it is straightforward to compute the distribution of the overall mean output $M = \frac{1}{N}\sum_{i=1}^{N} m(\mathbf{x}_i)$, in which N is the number of data points, $\mathbf{x}_i$ the ith data point and $m(\cdot)$ the emulator mean.

The results are shown in Figures 7.6 and 7.7. Error bars are given for the estimated mean and empirical 95% confidence interval, which are barely visible. They do not need to overlap because they are means and standard deviations for

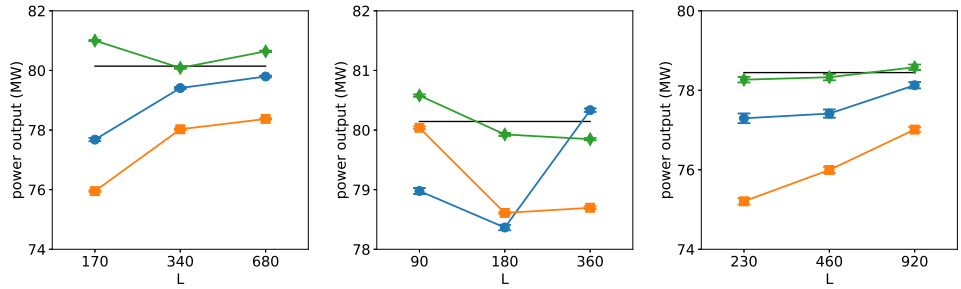


Figure 7.6: Results for the power output based on emulation. From left to right, data set 1 to 3 are presented. The legend can be found in Figure 7.4.

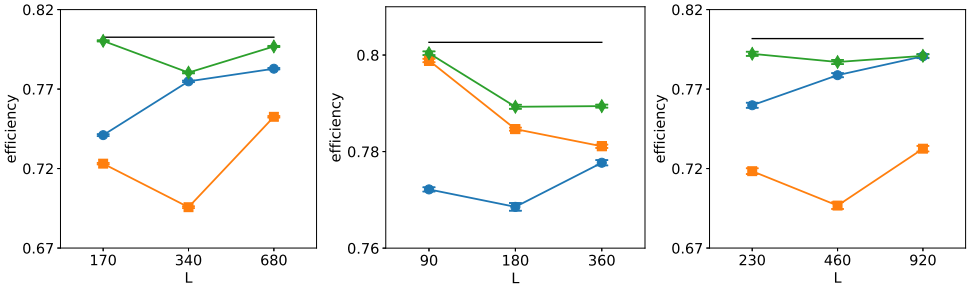


Figure 7.7: Results for the efficiency based on emulation. From left to right, data set 1 to 3 are presented. The legend can be found in Figure 7.4.

different emulators (although the emulators model the same system). The results for the power output are consistent with the results in Figures 7.4 and of comparable accuracy. The results for the efficiency are less accurate than the results in Figure 7.5. Also, most Gaussian processes underestimate the output. We note that because of their Gaussian nature, the emulators may occasionally predict a negative value of the output. This is the case for 4.5% of the data points when the power output is studied and for 4.2% of the data points when the efficiency is studied. We recomputed the average output in which the negative values were replaced by zero. This did not noticeably change the results.

Finally, we examine the quality of the Gaussian process for data set 2 with the k-means selected samples by k-fold cross-validation with $k = 10$ in Figures 7.8 and 7.9. We selected data set 2 because it does not include redundant input variables, which makes both the sample selection and the optimization problem for the emulator easier. Note the emulator often predicts a positive efficiency when the real efficiency is zero, which means it over-predicts in these cases, although this is not visible in the overall results. The value of $1-Q^2$ (5.10) for the power output (with $L \in \{90, 180, 360\}$) is given by $\{0.014, 0.020, 0.008\}$, while it

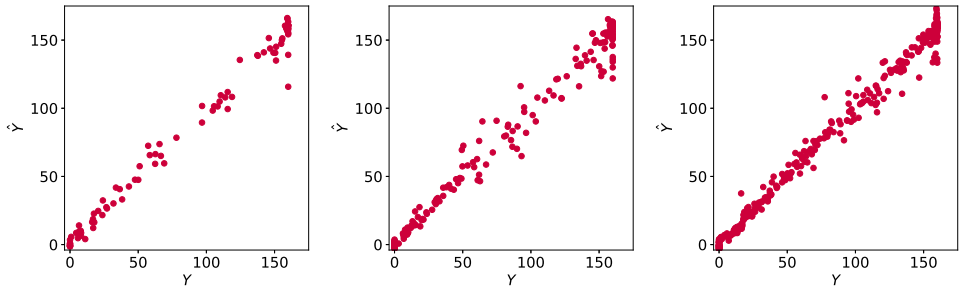


Figure 7.8: Cross-validation results for the power output (in MW). From left to right, $L \in \{90, 180, 360\}$ is presented.

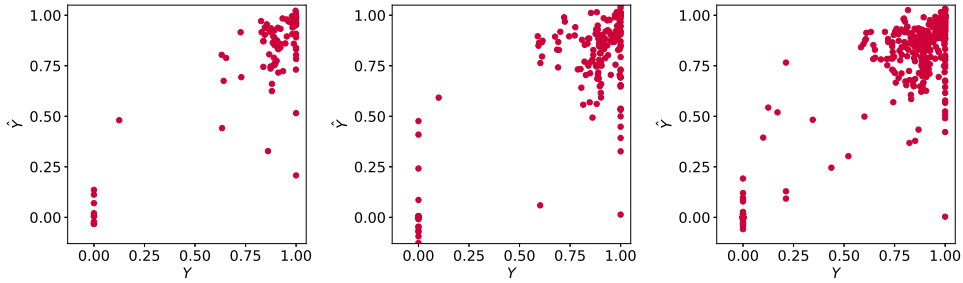
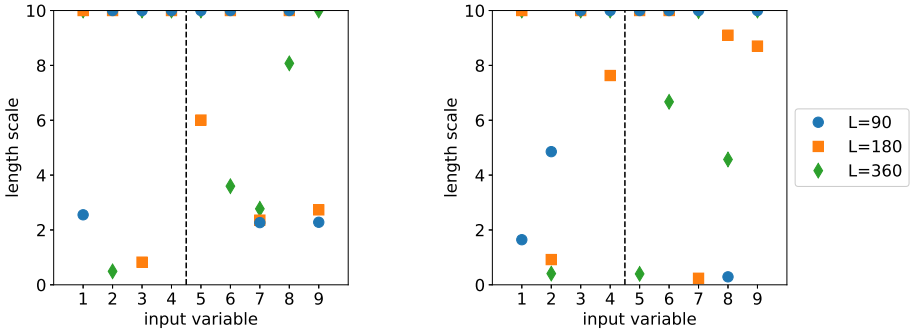


Figure 7.9: Cross-validation results for the efficiency. From left to right, $L \in \{90, 180, 360\}$ is presented.

is $\{0.255, 0.374, 0.207\}$ for the efficiency. On the basis of these results, the Gaussian processes for the efficiency are not accurate enough to perform sensitivity analysis on, while the ones for the power output are. Although a value of 0.01 should result in a "good" Gaussian process, this may be far from achievable in practice, and values near 0.01 can be accepted as well.

We go a little more into detail by studying the corresponding length scales of the Gaussian processes in Figure 7.10. The first four variables are the wind directions and the final five are the wind speeds; they are separated for clarity. We first look at the length scales for the power output in Figure 7.10a. One can see for the wind direction, only one of the wind directions is given a nontrivial length scale for each of the choices of L. For the wind speeds, two or three out of five have a nontrivial length scale, in which one of the wind speeds near hub height is always selected as nontrivial. Considering the length scales for the efficiency in Figure 7.10b, one or two wind directions have nontrivial length scales, and one or three wind speeds have nontrivial length scales.

This behavior may be due to the strong dependencies between them, and one can see this in the following way: both the wind directions and wind speeds are



(a) Power output. (b) Efficiency.

Figure 7.10: Estimated length scales for data set 2 with KME samples.

strongly dependent within their group of variables. Having one nontrivial length scale per group may be enough to capture the behavior of the complete group.

7.3.7 Sensitivity analysis

We continue with performing a sensitivity analysis on the cases for which the quality of the emulator was examined and turned out to be reasonably sufficient. This means we will perform a sensitivity analysis for the power output on data set 2, in which the Gaussian process is based on samples selected by k-means. For these settings, we found earlier $1 - Q^2 < 0.02$. In this case, we expect all variables to be influential. We have three emulators, based on a different number of samples L.

The results for the total sensitivity indices are given in Figure 7.11. The most influential input variables are the wind speeds at 85 and 92 m height, followed by the other wind speeds. Note the largest part of the rotor area is around hub height (70 m) and the wind speeds at different heights are strongly dependent. It is atypical that the sensitivity indices for wind direction are near zero, however, one can argue they are more important in the case of the efficiency, because the wind direction determines the strength of wake effects. The results from the different emulators are consistent in their assessment of the sensitivity indices.

We would like to compute the sensitivity indices for data set 3 as well, especially to compare the total and direct sensitivity indices. However, we have no validated Gaussian process for data set 3. Since data set 3 contains all the variables of data set 2, we can use the emulator of data set 2 (obtained with $L = 360$). To do this, we first scaled the length scales from the scaled domain of data set 2 to the scaled domain of data set 3, after which we predicted the power output by an emulator based on the length scales of data set 2. In these predictions, only

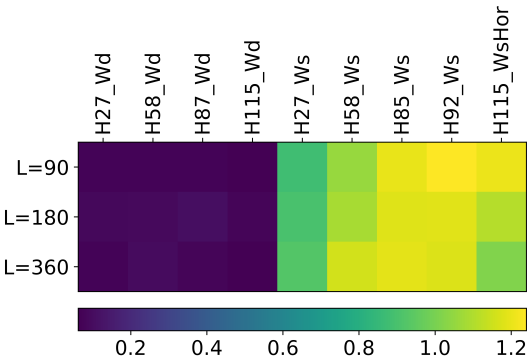


Figure 7.11: Results of the sensitivity analysis. Lighter colors indicate stronger sensitivities (see also Figure 7.2).

the first 9 variables are used. Then, we compute the total sensitivity indices and the direct sensitivity indices, in which we have applied the discreteness correction for the wave period variables. The results of the sensitivity analysis are in Figure 7.12. Here we see only the wind speed at 58 and 85 meters height has a clear direct effect, while the wind speed at 92 meters height has a very small direct effect. These are also the only wind speeds in the range of the rotor diameter. From these results, it is clear that the dependencies between the wind variables are important and lead to high values of the total sensitivity index. The sensitivity of the wave heights is shown to be indirect, which is due to their dependency with wind speed. This shows the direct sensitivity indices work as expected.

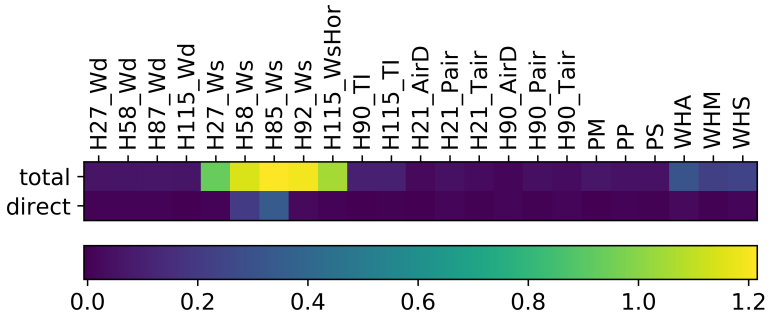
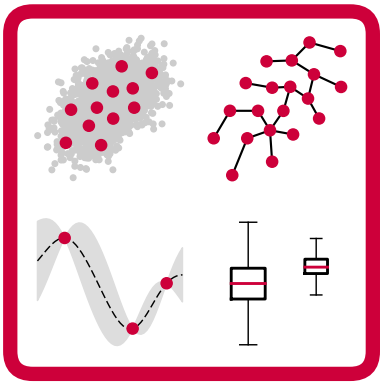


Figure 7.12: Sensitivity indices for the MMIJ test case.

7.4 Conclusion

We have shown how to apply the proposed framework in a real-life application which consists of the combination of a data set containing weather data and a computational model of power output for the Horns Rev wind farm. The framework correctly identifies dependencies in the data. We constructed several emulators and performed sensitivity analysis using them. The sensitivity analysis results are consistent over the different emulators considered.

It is of the utmost importance to validate the Gaussian process emulators before using them for predictions, because the predictions may be inaccurate if the Gaussian process did not perform well in validation tests. Another point of interest is the nature of the input variables, which can be periodic or discrete (intrinsic or due to measurement accuracy).



8 Case study II

The previous chapter was devoted to a case study which entailed the computation of the power output of a hypothetical offshore wind farm (OWF) and highlighted some of the problems one may encounter in practice. In this chapter, we will show a less straightforward application of the framework and illustrate the computational gains that can be achieved by it.

8.1 Introduction

The objective of the case study is to study in more detail the energy yield of a hypothetical offshore wind farm: again with the layout of Horns Rev I, but now located at Lichteiland Goeree (measurement station 320 of the Royal Netherlands Meteorological Institute) in the North Sea over the years 1981-2018. This energy yield is estimated not only for the complete data, but also for smaller time intervals. In this way, the energy yield per year or month can be obtained, and trends can be detected. This study is motivated by measurement campaigns held to calculate the expected revenues of future OWFs. These campaigns are limited in time, especially with respect to the lifespan of the OWFs. The inter-annual [136] and inter-seasonal [137] variability of the wind energy are then not taken into account properly. Hence, a measurement campaign with a duration of one year or even less may be insufficient and lead to either losses or unexpected higher profit.

A secondary objective is to show the computational gains achieved by using the framework instead of direct simulation of the available input data. We show the complete uncertainty quantification can be accelerated by two or three orders of magnitude, which makes it possible to obtain the power output, or a prediction thereof, for each available input data point. An extra benefit of the framework are the confidence intervals offered by the method at no extra cost.

The data we use contains i) wind speed at a height of 10m, or measured at a different height and converted to the equivalent wind speed at 10m, ii) mean

wind direction, (iii) time stamps and (iv) other weather data. Unfortunately, no vertical wind profiles are available, however the use of wind profiles has been discussed already in Chapter 7. The other weather data is only used in the first step of the analysis.

Section 8.2 describes the data analysis and necessary adaptations made to the raw data. Then, Section 8.3 shows the dependency analysis. The results of the study are shown in Section 8.4, where we show the expected energy yield over the years, computed with different methods. Section 8.5 describes possible financial implications, while Section 8.6 concludes.

8.2 Data analysis

We start by analyzing the data, which is done in two parts. First, we analyze the data necessary for the simulations, which is three-dimensional: two input variables which are used for the simulations itself and one time variable which is used as a label to create subsets of the data set. Then, we repeat the process for the data used in the dependency analysis, which is higher-dimensional.

8.2.1 For the simulations

We start by importing all available data and leave out the observations which do not have a time stamp, wind direction or wind speed. Also, we remove the observations for which the wind direction varied during the measurement period (which is one hour). These observations are less than half a percent of the data. Then, we transform the time stamp to a year, because day and month are irrelevant for our current purpose. Furthermore, we transform the wind speed to m/s (from 0.1 m/s). We list the unique values of wind speed and direction to look for outliers and we find one, which we remove because its value is outside the range of admitted values. This leaves us with 274191 observations.

We continue with checking the fraction of time for which data is available (availability) by looking at the number of complete, included observations per year in Figure 8.1. In this figure, we have also drawn the line of 99% availability of data (no correction for leap years in the figure). To avoid seasonal effects, we now remove the years below this threshold, being 1981-1993 and 2001-2003. This leaves $N = 183768$ included observations.

We show the remaining distributions for wind direction and wind speed in Figure 8.2. First, we note in Figure 8.2a both 0 and 360 are included. Here, 360 indicates north, while 0 indicates calm conditions and therefore no measurable wind direction. Second, we note the bars in Figure 8.2b are not uniformly distributed. This is caused on the one hand by recent observations being measured

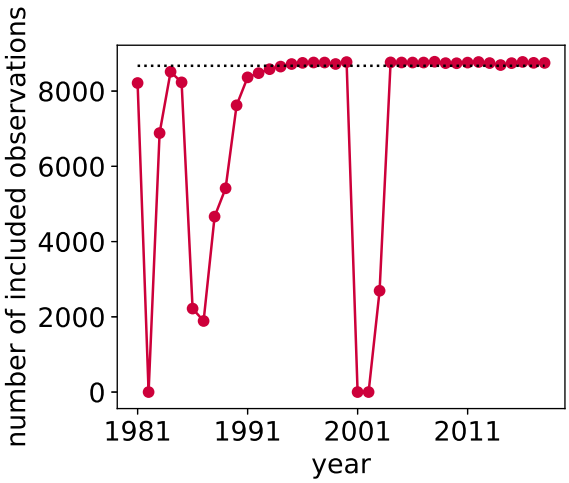
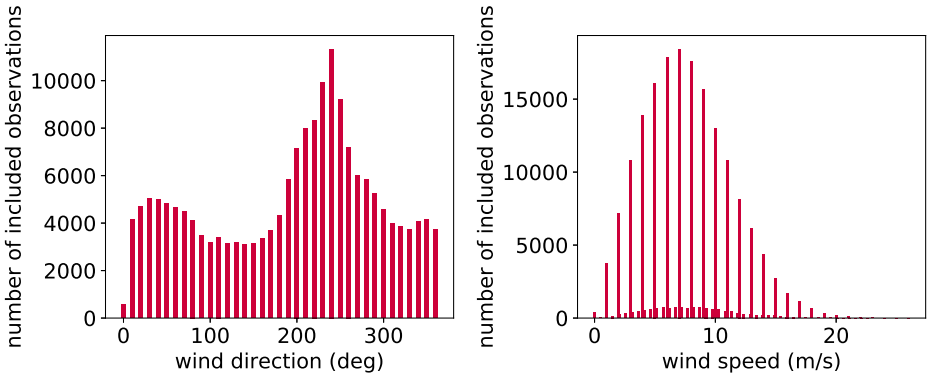


Figure 8.1: Number of observations per year.



(a) Wind direction.

(b) Wind speed.

Figure 8.2: Barplots of the (discrete) observations of wind direction and wind speed.

in integer multiples of 1 m/s rather than fractions thereof, and on the other hand by a conversion of older data.

We combine wind direction and wind speed measurements in a wind rose plot. The wind rose of all included data is given in Figure 8.3. This should be interpreted as a periodic version of a stacked bar diagram, which is normalized to represent a probability distribution function. The color indicates the wind speed. The number of sectors is 36, such that each value of the wind direction has its own bar. The measurements with calm conditions are not included. One can clearly see the prevailing wind direction in the Netherlands is southwest.

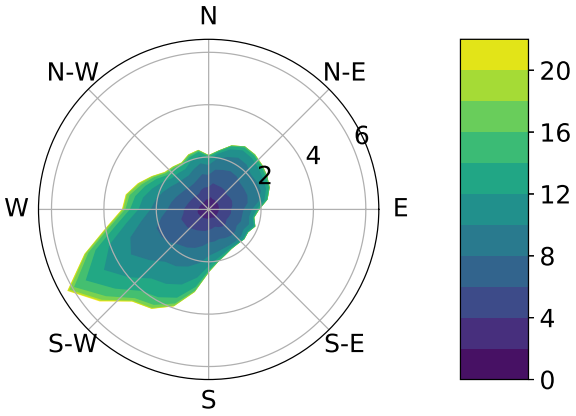


Figure 8.3: Wind rose.

8.2.2 For the dependency analysis

For the dependency analysis, we used the data consisting of the variables given in Table 2.2, for which we included the time stamp in order to select the data obtained in the years 1981-2018. Then, we removed incomplete observations, the calm and variable wind measurements, and the outlier measurement found earlier. No other outliers were found. In this way, we have 209915 observations.

We check the availability of the data in Figure 8.4, in which we also plotted the 99% availability line, although this is almost never reached. Because we only use this data to demonstrate the dependencies, we accept this distribution of data over the years.

8.3 Dependency analysis

The results of the dependency analysis are shown in Figure 8.5a, for which we computed $D_\alpha(g, \tilde{g})$ (2.5). These are very similar to the ones in Figure 2.15. The labels are explained in Table 2.2. We have also implemented a discreteness correction, of which the results are in Figure 8.5b. This correction is similar to the one used in Chapter 7 for the wave periods, although we applied it here to all variables. One can see the values of the dependency index have decreased by a large extent. Dependencies between the wind speed variables still exist, just as between the temperature variables, between wind speeds and visibility and between visibility and relative atmospheric humidity. The idea behind the discreteness correction is to make the marginal distributions more uniform, such that they are not reflected in the dependency index. However, by doing this, some parts of the dependency may be destroyed. This can be understood as follows: by the

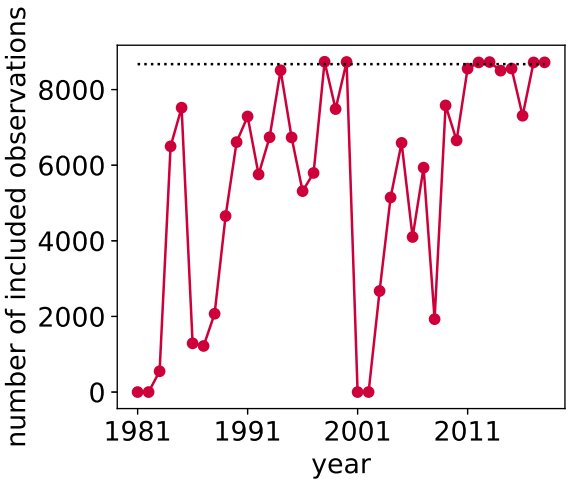
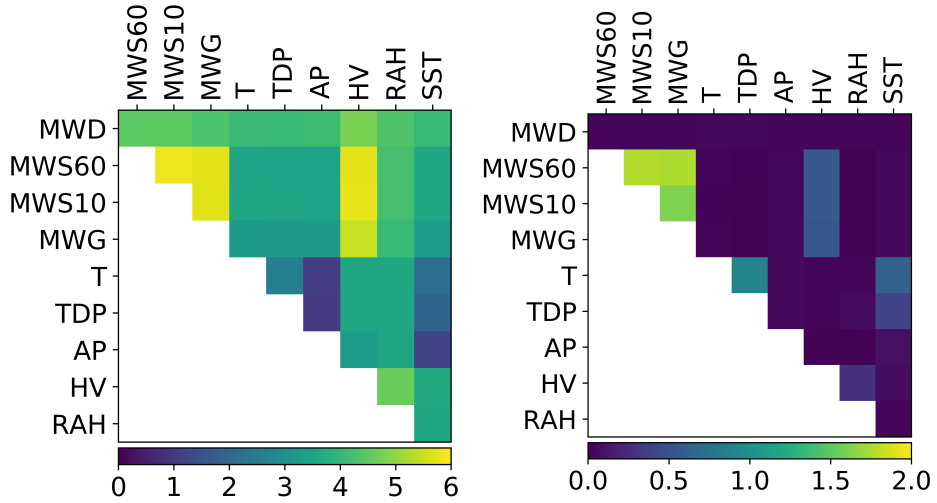


Figure 8.4: Number of complete, valid observations of the data set per year.



(a) Original.

(b) With discreteness correction.

Figure 8.5: Dependency analysis, without (left) and with (right) discreteness correction.

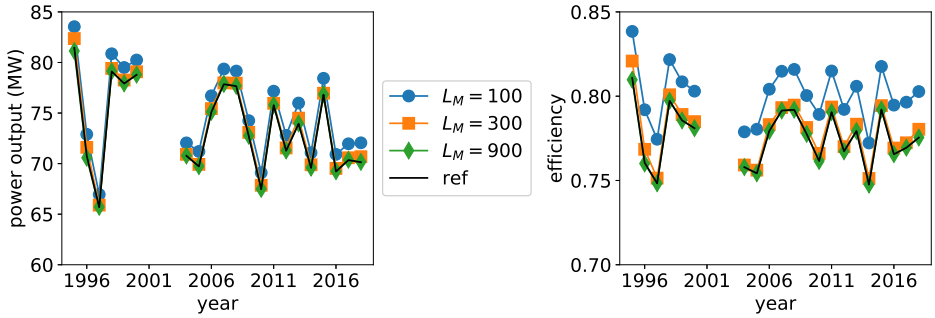
discretization, several data points get the same value for two variables. In the original rank-transform, they are placed at exactly the same location after which a small noise is applied to separate them numerically. If any dependency between them existed, this is not seen by the aggregation due to the measurement accuracy. Hence, applying the discreteness correction might actually underestimate

the dependencies. However, without discreteness correction, the dependencies are overestimated because the marginal distributions can be far from uniform when the discretization is coarse. At the moment, the only remedy would be to have more precise measurements.

8.4 Results

We now return to the original goal of the case study, which is to study the energy yield of a hypothetical OWF. The idea is to study the variation in energy yield over the years and to show the computational gains achieved by using the framework.

We assume we are allowed to use no more than $L = 10^3$ evaluations of the computational model because of computational cost constraints. Of these, we reserve $L_V = 100$ for the validation and use the remaining $L_M = 900$ for their output and as basis for the emulator. The L_M samples are selected by KME and have been evaluated. We will also use lower values of L_M to see if using less samples is sufficient as well. From the obtained samples, we compute the quantities of interest by the piece-wise constant approximation on the clusters. This is done per year, in which the weights are computed from the relevant input data. The results can be found in Figure 8.6. For reference purposes, we have also computed the power output for each data point individually. In practice, this will not be possible.



(a) Power output.

(b) Efficiency.

Figure 8.6: Computed output for the years in which enough data is present, based on weighted samples.

Figure 8.6 shows there exists quite some variation over the different years, although no trend is visible. We investigate this result further in Section 8.5. Furthermore, we see using $L_M = 100$ samples overestimates the power output and the efficiency, while $L_M = 300$ seems sufficient. This means only $300 + 100$ evaluations of the computational model are sufficient to replace the 183768 original

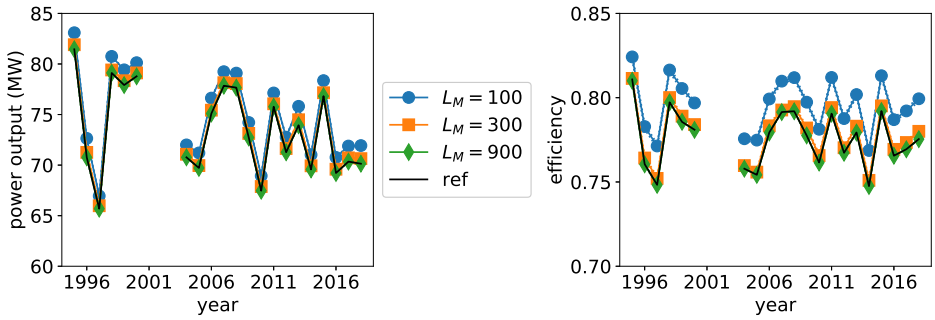
evaluations, which is a reduction between 2 and 3 orders of magnitude. The computational time to obtain these samples using KME is negligible with respect to the time needed by the computational model.

We continue by constructing a Gaussian process on the L_M samples. To validate the Gaussian process, the L_V samples have been selected by stratified sampling in time of the data. Several other choices for selecting the validation data are possible as well, but in this way we expect to get a good coverage of the input data.

The validation results are given by $1-Q^2 = \{8.0 \cdot 10^{-3}, 1.3 \cdot 10^{-3}, 1.5 \cdot 10^{-6}\}$ for the power output (with $L_M = \{100, 300, 900\}$) and $1-Q^2 = \{5.4 \cdot 10^{-2}, 2.5 \cdot 10^{-2}, 5.4 \cdot 10^{-5}\}$ for the efficiency, which means the plot of the emulated versus the real output values is very close to a straight line in almost all cases. Sometimes, the power output or efficiency is estimated below zero, but this has only a very minor effect. Due to these validation results, we are confident enough to use the prediction data for the complete input data already with $L_M = 100$ when it concerns the power data, and with $L_M = 900$ also for the efficiency. In a moment, we will see this is not ideal due to the bias for $L_M = 100$.

We show the results of the power output based on the predictions together with a 95% confidence interval in Figure 8.7. The difference between Figures 8.6 and 8.7 is the prediction method: the first predicts for all data points in the cluster the output value of the cluster center, while the second uses a Gaussian process to predict the output values. The confidence intervals are computed from the prediction variance at each prediction location. They are not visible at all due to their small width. We see again the results for $L = 100$ over-predict, while the other results are very accurate in the estimation mean, especially for the power output. This over-prediction is consistent with the computed output based on weighted samples which also over-predicts for $L = 100$. It does not contradict the result of the validation which measures precision rather than accuracy. However, it shows emulators are not perfect and their quality depends on the samples they are based on. Note the computed output based on weighted samples also over-predicts for $L = 100$. The small width of the confidence intervals is caused by the large amount of samples, as the standard deviation of the mean decreases with increasing N. From the figure, we conclude $L = 300$ samples are enough to properly capture the features of the energy yield. Also, we do not see any trend in the energy yield and the variations per year can be large.

The cost of constructing the emulator and obtaining its predictions is small with respect to the computational time of the model. The cost of obtaining the predictions is only the cost of solving a linear system, which is negligible with respect to the cost of the parameter optimization of the Gaussian process. When more input variables are included, the construction time will increase.



(a) Power output.

(b) Efficiency.

Figure 8.7: Computed output for the years in which enough data is present, based on Gaussian processes.

One can summarize this time gain as follows: instead of performing N simulations with total time cost $N \cdot T_{sim}$, one only performs $L_M + L_V$ simulations at cost $(L_M + L_V) \cdot T_{sim}$, plus the time to obtain the predictions $T_{pred} = T_{pred}(L_M, N)$, which depends in practice on the dimension of the problem as well. In practice, where $N \gg L_M$, L_M not larger than 10^3 and $T_{sim} \gg T_{pred}$, this leads to a large computational speedup.

8.5 Indication of financial consequences

In this section, we illustrate the profitability of such a wind farm under strongly simplifying assumptions regarding, e.g., costs and operations. We use the data from [138] regarding the Horns Rev I wind farm. We fix all costs and look only at the effect of the gross annual energy production. Because of the small confidence intervals, we focus on the mean value. For each year, we compute whether this wind farm would have been profitable for the given parameter values and input data. Note the only differences between our case and [138] are the data used, which is here obtained from the Lichteiland Goeree, about 500 km away from the original wind farm, and the computational model to obtain the power output.

The results are in Figure 8.8. In [138], the selling price of the energy is 0.11 euro per kWh, which means the levelized cost-of-energy should stay below this value to be profitable. One can see the profitability of this OWF is limited, and depending on the chosen period for data collection, it would be considered profitable or not. This is not a desired situation. This issue can be solved by making use of a wind atlas [139] or reanalysis data such as ERA5 [140], because it contains wind data for longer periods of time.

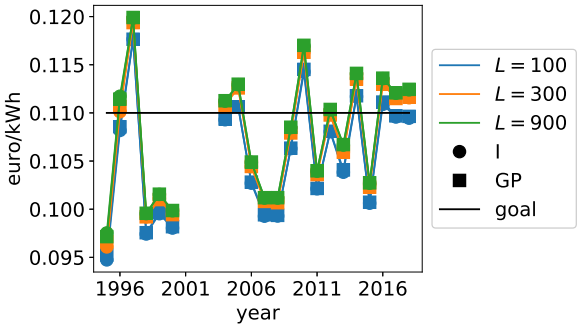
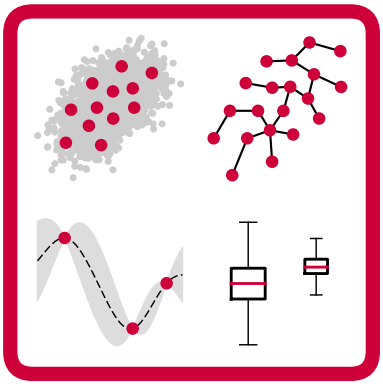


Figure 8.8: Profitability.

8.6 Conclusion

We have shown the proposed framework can also be applied to perform UQ in a different setting: here, we performed UQ on parts of the input data. In this case, the goal was to investigate the behavior of the energy yield over the years, which was shown to be highly variable.

Furthermore, we have illustrated the computational gain which can be achieved by using the framework rather than full simulation of the available data. In this way, expensive computational models can be used in uncertainty quantification, which is expected to lead to improved and novel results.



9 Conclusion

The goal of this chapter is to summarize the obtained results and to look forward to possible extensions and improvements.

9.1 Conclusion

We studied different aspects of uncertainty quantification (UQ) with the focus on the case of dependent input variables available in the form of data. The need for uncertainty quantification is usually caused by the complexity of the system under consideration, which involves input, a process and the output. The process is often an expensive computational model of several physical processes, such that not all desired simulations can be performed due to the complexity of the model and the resulting computational cost. The challenge is to select suitable samples from the input data which are used to perform the simulations. This is the step at the heart of the UQ process. At the beginning of the process, the dependencies present in the data can be studied to give more insight in the system. After performing the chosen simulations, one can continue the UQ process by predicting the process output for other input data and performing a sensitivity analysis. This leads to a complete framework of which the separate steps are detailed below.

In Chapter 2, we developed an efficient approach for quantifying dependencies in data by means of Rényi mutual information. This approach is mainly applicable to large multivariate data sets. We also proposed a novel estimator in case the ranking of the dependencies is required instead of their exact strength. The method gives consistent results and is computationally feasible due to the use of minimum spanning trees.

The main topic of Chapter 3 was to construct a sample selection method which works well for dependent input variables. We achieved this goal by employing clustering techniques, especially k-means clustering and using the cluster centers as samples. A good representation of the input data can be achieved with relatively few samples, especially when the input data is strongly dependent.

Chapter 4 considered emulation and especially Gaussian processes which is an often-used concept in emulation. Furthermore, we derived a theoretical result on upper bounds for the prediction variance of Gaussian processes. This chapter also served as a preliminary for the following chapters, in which Gaussian processes are frequently used as emulator.

The goal of Chapter 5 was to improve the estimates of divergence-based sensitivity indices by including combinations of input data and output predicted by an emulator. This is advantageous in the case where the number of available samples is small. The method can be used with the same estimation mechanism as Chapter 2 or with kernel density estimation. Furthermore, we developed an extra type of sensitivity indices to distinguish between direct and indirect sensitivity effects.

Chapters 2, 3 and 5 contain the main parts of the UQ framework. Chapter 6 contains a possible extension, namely a way to add extra samples to an existing sampling design. In the test cases in Chapter 6, the method does not outperform sampling designs in which all the samples are generated in one step. However, it also does not perform worse, and can be beneficial if not all simulation efforts can be used immediately.

Finally, Chapters 7 and 8 considered two applications of the framework in the domain of offshore wind energy. The first application was a straightforward UQ of a computational model of the power output of an offshore wind farm, in which the input data was a data set containing weather data. The second application had a more specific goal, namely determining whether a possible setup of an offshore wind farm would be profitable. The focus was on the year-to-year variability of the energy yield.

In general, the proposed framework works well and can be applied in practice. However, there are a few points of interest and improvements are certainly possible. Some of them will be explained in more detail in the next section.

9.2 Suggestions for further research

We will first give suggestions per step of the UQ framework, after which we indicate general suggestions.

9.2.1 Dependency analysis

In some of the test cases, groups of dependent variables appeared. It would be useful to be able to characterize and visualize them as such, and even compute dependency indices between groups. This would require a generalization of the rank-transform which not only removes the marginal distribution, but also takes care of the dependencies within a group. This is a higher-dimensional version of

the problem encountered before, namely, how to remove the effect of the marginal distributions from the dependency index.

Another issue still open for further research is the presence of periodic variables. One or two of them per data set may be handled by adapting the distance function, but this becomes cumbersome in the case of many periodic variables. A comparable issue is the presence of discrete variables (truly discrete or by measurement accuracy), which is currently handled by either placing them at the same location in the rank-transform (with a small noise term to avoid problems with the minimum spanning tree) or by randomly assigning them adjoining locations. Both are technically incorrect and suffer from (different) problems.

9.2.2 Sample selection

As indicated before, the current method for sample selection as described in Chapter 3 is only first-order accurate, as only constants and linear functions are integrated correctly. Extensions to higher-order accurate can be developed in several ways. We illustrate one possible method in which the sample locations stay the same, but the weights are adapted. It is possible to construct a linear system containing the constant term, linear terms and quadratic terms (mixed or from one variable), in which the weights are the variables to be solved for. Because the number of data points is much larger than the number of samples, this system is under-determined and leaves space for the possibility of optimization in the solution space. In this solution space, one solves an optimization problem which minimizes the norm between the original clustering weights and the weights obtained from the linear system [141].

A second option to obtain higher-order accurate methods is to construct hybrid methods which combine the proposed sample selection methods with regular UQ sample selection methods. In this way, one could handle the dependencies and get the higher-order accuracy from the regular methods.

9.2.3 Emulation

We use log-likelihood optimization to obtain the length scales for the Gaussian process. Another option would be to obtain the length scales via Bayesian inference. [4] already used this to obtain the distribution and density functions of the output given the input. This adaptation could be advantageous when the number of samples L or the input dimension d is large.

9.2.4 Sensitivity analysis

The issues appearing in sensitivity analysis are from a different kind than the ones in the dependency analysis, although their methods are very similar. We have already developed direct and total sensitivity indices, but when one thinks of Sobol's indices, also higher-order sensitivity indices (for mixed terms) appear, which are combined into the total sensitivity indices. An interesting research topic would be to derive higher-order sensitivity indices and a suitable estimation method along the lines of the approach in Chapters 2 and 5.

9.2.5 General

In this section, we discuss two options: adaptive sampling and dimension reduction.

The idea behind both options is to make better use of the information available from the UQ process. We have shown a possible option for adaptive sampling in Chapter 6, but this leaves room for further development and improvement. In this method, the available information comes from the samples and their corresponding model output.

The dimension reduction can be performed at several steps in the process. When it is known that some variables are not influential or even not used in obtaining the output, one can remove them from the probability space and replace them with a fixed value. In general, one can also apply a dimension reduction after the dependency analysis. When a group of variables is strongly dependent, one can select the variable with strongest dependencies to the others, or try to fit a lower-dimensional manifold from which samples are selected.

As a final remark, one is free to change the settings of different methods in order to get better results for a specific problem. These settings include, but are not limited to the clustering method, Gaussian process kernel, distance function of the Gaussian process and the type of divergence of the sensitivity index. One can even go further and replace complete parts of the framework when better methods are developed in the future.

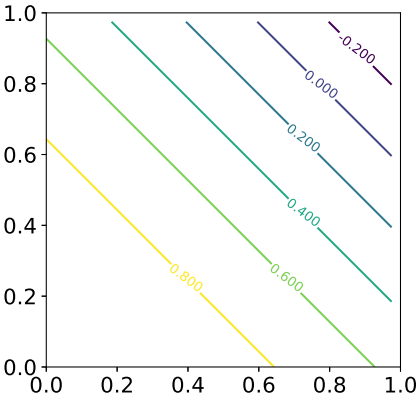
A Genz test functions

In Table A.1, the definitions of the Genz functions are given. The parameters **a** can be used to make the function harder or easier to integrate, while **u** contains scale parameters. The functions are defined in d dimensions, in which $d \in \mathbb{N}$, on the domain $[0,1]^d$. In all tests, we will choose $a_i = 1$ for $i = 1,\ldots,d$. We will choose $u_i = 1/2$ for $i = 1,\ldots,d$ for all functions except for f_1, where we choose $u_1 = 0$.

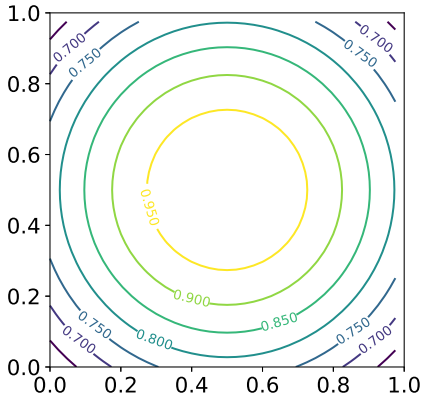
Table A.1: Definition of the Genz test functions.

Nr	Characteristic	Function
1	Oscillatory	$f_1(\mathbf{x}) = \cos\left(2\pi u_1 + \sum_{j=1}^{d} a_j x_j\right)$
2	Gaussian peak	$f_2(\mathbf{x}) = \exp\left(-\sum_{j=1}^{d} a_j^2 (x_j - u_j)^2\right)$
3	C_0	$f_3(\mathbf{x}) = \exp\left(-\sum_{j=1}^{d} a_j \lvert x_j - u_j\rvert\right)$
4	Product peak	$f_4(\mathbf{x}) = \prod_{j=1}^{d} \left(a_j^{-2} + (x_j - u_j)^2\right)^{-1}$
5	Corner peak	$f_5(\mathbf{x}) = \left(1 + \sum_{j=1}^{d} a_j x_j\right)^{-d+1}$
6	Discontinuous	$f_6(\mathbf{x}) = \begin{cases} 0 & x_1 > u_1 \quad \text{or} \quad x_2 > u_2 \\ \exp\left(\sum_{j=1}^{d} a_j x_j\right) & \text{else} \end{cases}$

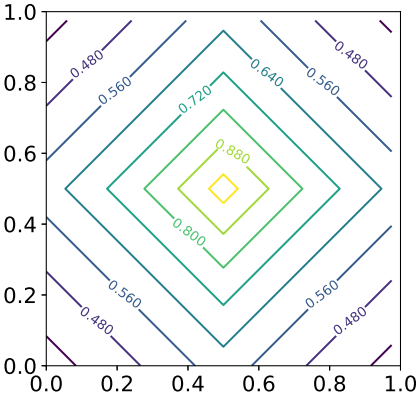
In Figure A.1, the values for the Genz functions on the domain $[0,1]^2$ are visualized. Test functions 2 and 4 look similar, but are different.



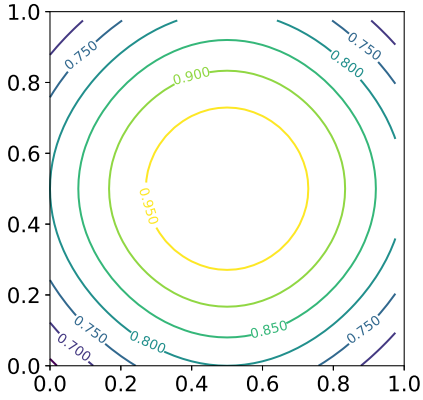
(a) Test function 1.



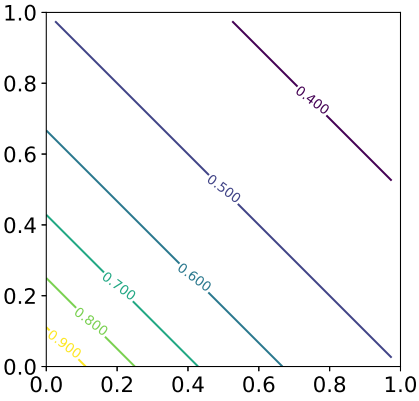
(b) Test function 2.



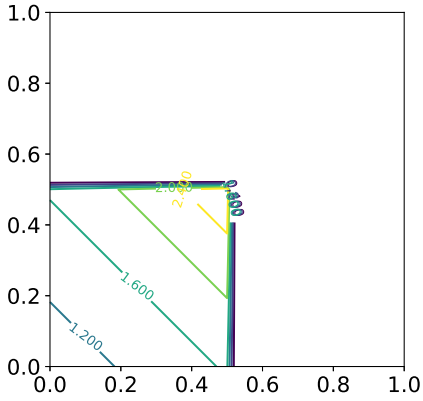
(c) Test function 3.



(d) Test function 4.



(e) Test function 5.



(f) Test function 6.

Figure A.1: Genz test functions.

B Meteomast IJmuiden variables

The variables of included in the analysis are given in Table B.1. The names contain (if applicable) the height at which the variables are recorded (Hx for x m), the quantity measured (Wd for wind direction, Ws for wind speed, etc.), and in some cases additional information (e.g. signal quality). See [15] for more details.

Table B.1: Explanation of the included variables.

Variable name	Measurement	Unit
MMIJ_Hx_Wd_Q1_avg	Wind direction at x m	m/s
MMIJ_ZPHS_H115_Wd	Wind direction at 115 m	deg
MMIJ_Hx_Ws_Q1_avg	Wind speed at x m	m/s
MMIJ_ZPHS_H115_WsHor_avg	Horizontal wind speed at 115 m	m/s
MMIJ_ZPHS_Hx_TI	Turbulence intensity at x m	-
MMIJ_Hx_AirDensity_Q1_avg	Air density at x m	-
MMIJ_Hx_Pair_Q1_avg	Air pressure at x m	hPa
MMIJ_Hx_Tair_Q1_avg	Air temperature at x m	deg C
MMIJ_BW_PeriodMean	Mean wave period	s
MMIJ_BW_PeriodPeak	Peak wave period	s
MMIJ_BW_WavePeriodSig	Significant wave period	s
MMIJ_BW_WaveHeightAvg	Average wave height	m
MMIJ_BW_WaveHeightMax	Maximum wave height	m
MMIJ_BW_WaveHeightSig	Significant wave height	m

Their inclusion in the three data sets are given in Table B.2.

Table B.2: Variables of the MMIJ included in the analysis, see also [15].

Variables in sets 1,2,3	Variables in sets 1,3	Variables in set 3
MMIJ_H27_Wd_Q1_avg	MMIJ_ZPHS_H90_TI	MMIJ_BW_PeriodMean
MMIJ_H58_Wd_Q1_avg	MMIJ_ZPHS_H115_TI	MMIJ_BW_PeriodPeak
MMIJ_H87_Wd_Q1_avg	MMIJ_H21_AirDensity_Q1_avg	MMIJ_BW_WavePeriodSig
MMIJ_ZPHS_H115_Wd	MMIJ_H21_Pair_Q1_avg	MMIJ_BW_WaveHeightAvg
MMIJ_H27_Ws_Q1_avg	MMIJ_H21_Tair_Q1_avg	MMIJ_BW_WaveHeightMax
MMIJ_H58_Ws_Q1_avg	MMIJ_H90_AirDensity_Q1_avg	MMIJ_BW_WaveHeightSig
MMIJ_H85_Ws_Q1_avg	MMIJ_H90_Pair_Q1_avg	
MMIJ_H92_Ws_Q1_avg	MMIJ_H90_Tair_Q1_avg	
MMIJ_ZPHS_H115_WsHor_avg		

List of abbreviations

CDF ..... cumulative distribution function

FMST ..... fast minimum spanning tree

GP ..... Gaussian process

KDE ..... kernel density estimator/estimate/estimation

KL ..... Kullback-Leibler

KME ..... *k*-means

LHD ..... Latin hypercube design

LHS ..... Latin hypercube sampling

MCC ..... Monte Carlo clustering

MC,MCS ..... Monte Carlo sampling

MMIJ ..... Meteomast IJmuiden

MST ..... minimum spanning tree

OWF ..... offshore wind farm

PCA ..... principal component analysis

PCR ..... reassigned principal component analysis

PDD ..... polynomial dimensional decomposition

SOS ..... sum-of-squares

UQ ..... uncertainty quantification

List of publications

1. A. Eggels, R. Kunnen, B. Koren, and A. Tijsseling, "Infotaxis in a turbulent 3D channel flow," *Journal of Computational and Applied Mathematics*, vol. 310, pp. 44–58, 2017.

   This publication is not part of the thesis.

2. A. Eggels, D. Crommelin, and J. Witteveen, "Clustering-based collocation for uncertainty propagation with multivariate dependent inputs," *International Journal for Uncertainty Quantification*, vol. 8, no. 1, pp. 43–59, 2018.

   Chapter 3 is based on this research article. The research work, numerical simulations and writing the article were performed by A.W. Eggels. The research work was carried out under the supervision of D. Crommelin and J. Witteveen. J. Witteveen initiated the research topic.

3. A. Eggels and D. Crommelin, "Uncertainty Quantification with dependent inputs: wind and waves," in *Proceedings of ECCM 6 and ECFD 7*, 2018.

   Chapter 7 is based on this research article. The research work, numerical simulations and writing the article were performed by A.W. Eggels. The research work was carried out under the supervision of D. Crommelin.

4. A. Eggels and D. Crommelin, "Quantifying Data Dependencies with Rényi Mutual Information and Minimum Spanning Trees," *Entropy*, vol. 21, no. 2, article no. 100, 2019.

   Chapter 2 is based on this research article. The research work, numerical simulations and writing the article were performed by A.W. Eggels. The research work was carried out under the supervision of D. Crommelin.

5. A. Eggels and D. Crommelin, "Efficient estimation of divergence-based sensitivity indices with Gaussian process surrogates," *in preparation*, 2019.

Chapter 5 is based on an early version of this research article. The research work, numerical simulations and writing the article were performed by A.W. Eggels. The research work was carried out under the supervision of D. Crommelin.

6. A. Eggels, "Uncertainty quantification with dependent input data - including applications to offshore wind farms," *Software published on Zenodo*, 10.5281/zenodo.3235924.

   This software is developed to obtain the results in this thesis and contains also the code to generate the results for the Ishigami example.

Bibliography

[1] Centraal Bureau voor de Statistiek, "Hernieuwbare energie in Nederland 2016," 2017.

[2] R. Postma, "Eerste windparken zonder subsidie," *NRC*, 2017-04-18.

[3] B. Zuidervaart, "Nederland heeft wereldprimeur: een windmolenpark zonder subsidie," *Trouw*, 2018-03-19.

[4] M. C. Kennedy and A. O'Hagan, "Bayesian calibration of computer models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 3, pp. 425–464, 2001.

[5] B. Iooss and P. Lemaître, "A review on global sensitivity analysis methods," in *Uncertainty Management in Simulation-Optimization of Complex Systems*, pp. 101–122, Springer, 2015.

[6] O. P. Le Maître and O. M. Knio, *Spectral Methods for Uncertainty Quantification: With Applications to Computational Fluid Dynamics*. Scientific Computation, Springer, 2010.

[7] D. Xiu, *Numerical Methods for Stochastic Computations: A Spectral Method Approach*. Princeton University Press, 2010.

[8] R. C. Smith, *Uncertainty Quantification: Theory, Implementation, and Applications*, vol. 12 of *Computational Science and Engineering*. SIAM, 2013.

[9] T. J. Sullivan, *Introduction to Uncertainty Quantification*, vol. 63 of *Texts in Applied Mathematics*. Springer, 2015.

[10] R. Ghanem, D. Higdon, and H. Owhadi, *Handbook of Uncertainty Quantification*. Springer, 2017.

[11] H. Bijl, D. Lucor, S. Mishra, and C. Schwab, *Uncertainty Quantification in Computational Fluid Dynamics*, vol. 92 of *Lecture Notes in Computational Science and Engineering*. Springer, 2013.

[12] J. M. Murphy, D. M. H. Sexton, D. N. Barnett, G. S. Jones, M. J. Webb, M. Collins, and D. A. Stainforth, "Quantification of modelling uncertainties in a large ensemble of climate change simulations," *Nature*, vol. 430, no. 7001, pp. 768–772, 2004.

[13] M. Grigoriu, *Stochastic systems: uncertainty quantification and propagation*. Springer Science & Business Media, 2012.

[14] W. Weibull, "A statistical distribution function of wide applicability," *Journal of Applied Mechanics*, vol. 18, no. 3, pp. 293–297, 1951.

[15] "Meteomast IJmuiden." `https://www.windopzee.net/meteomast-ijmuiden-mmij/introductie/index.html`. Accessed: 2019-05-22.

[16] M. Méchali, R. Barthelmie, S. Frandsen, L. Jensen, and P.-E. Réthoré, "Wake effects at Horns Rev and their influence on energy production," in *European Wind Energy Conference and exhibition 2006*, pp. 10–20, WindEurope, 2006.

[17] R. Barthelmie, S. Frandsen, K. Hansen, J. Schepers, K. Rados, W. Schlez, A. Neubert, L. Jensen, and S. Neckelmann, "Modelling the impact of wakes on power output at Nysted and Horns Rev," in *European Wind Energy Conference and exhibition 2009*, vol. 2, pp. 1351–1373, WindEurope, 2009.

[18] K. S. Hansen, R. J. Barthelmie, L. E. Jensen, and A. Sommer, "The impact of turbulence intensity and atmospheric stability on power deficits due to wind turbine wakes at Horns Rev wind farm," *Wind Energy*, vol. 15, no. 1, pp. 183–196, 2011.

[19] M. Gaumond, P.-E. Réthoré, S. Ott, A. Peña, A. Bechmann, and K. S. Hansen, "Evaluation of the wind direction uncertainty and its impact on wake modeling at the Horns Rev offshore wind farm," *Wind Energy*, vol. 17, no. 8, pp. 1169–1178, 2013.

[20] J. C. Helton, J. D. Johnson, C. J. Sallaberry, and C. B. Storlie, "Survey of sampling-based methods for uncertainty and sensitivity analysis," *Reliability Engineering & System Safety*, vol. 91, no. 10, pp. 1175–1209, 2006.

[21] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola, *Global Sensitivity Analysis. The Primer*. John Wiley & Sons, 2008.

[22] T. M. Cover and J. A. Thomas, *Elements of Information Theory,* Second Edition. John Wiley & Sons, 2006.

[23] A. O. Hero, B. Ma, O. J. J. Michel, and J. Gorman, "Applications of entropic spanning graphs," *IEEE signal processing magazine*, vol. 19, no. 5, pp. 85–95, 2002.

[24] A. O. Hero, J. Costa, and B. Ma, "Asymptotic relations between minimal graphs and α-entropy," Tech. Rep. 334, Comm. and Sig. Proc. Lab.(CSPL), Dept. EECS, University of Michigan, Ann Arbor, 2003.

[25] A. O. Hero and O. J. J. Michel, "Robust entropy estimation strategies based on edge weighted random graphs," in *SPIE's International Symposium on Optical Science, Engineering, and Instrumentation*, pp. 250–261, International Society for Optics and Photonics, 1998.

[26] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical Society*, vol. 7, no. 1, pp. 48–50, 1956.

[27] R. C. Prim, "Shortest connection networks and some generalizations," *Bell System Technical Journal*, vol. 36, no. 6, pp. 1389–1401, 1957.

[28] B. Auder and B. Iooss, "Global sensitivity analysis based on entropy," in *Safety, reliability and risk analysis-Proceedings of the ESREL 2008 Conference*, pp. 2107–2115, 2008.

[29] H. Liu, W. Chen, and A. Sudjianto, "Relative entropy based method for probabilistic sensitivity analysis in engineering design," *Journal of Mechanical Design*, vol. 128, no. 2, pp. 326–336, 2006.

[30] T. van Erven and P. Harremoes, "Rényi divergence and Kullback-Leibler divergence," *IEEE Transactions on Information Theory*, vol. 60, no. 7, pp. 3797–3820, 2014.

[31] D. Pál, B. Póczos, and C. Szepesvári, "Estimation of Rényi entropy and mutual information based on generalized nearest-neighbor graphs," in *Advances in Neural Information Processing Systems*, pp. 1849–1857, 2010.

[32] K. Moon, K. Sricharan, K. Greenewald, and A. Hero, "Ensemble estimation of information divergence," *Entropy*, vol. 20, no. 8, article no. 560, 2018.

[33] A. Hero and O. J. J. Michel, "Estimation of Rényi information divergence via pruned minimal spanning trees," in *Proceedings of the IEEE Signal Processing Workshop on Higher-Order Statistics*, pp. 264–268, IEEE, 1999.

[34] M. Rosenblatt, "Remarks on a multivariate transformation," *The Annals of Mathematical Statistics*, vol. 23, no. 3, pp. 470–472, 1952.

[35] E. Torre, S. Marelli, P. Embrechts, and B. Sudret, "A general framework for data-driven uncertainty quantification under complex input dependencies using vine copulas," *Probabilistic Engineering Mechanics*, vol. 55, pp. 1–16, 2019.

[36] W. J. Conover, "The rank transformation–an easy and intuitive way to connect many nonparametric methods to their parametric counterparts for seamless teaching introductory statistics courses," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 4, no. 5, pp. 432–438, 2012.

[37] C. Spearman, "The proof and measurement of association between two things," *The American Journal of Psychology*, vol. 15, no. 1, pp. 72–101, 1904.

[38] A. L. Gibbs and F. E. Su, "On choosing and bounding probability metrics," *International Statistical Review*, vol. 70, no. 3, pp. 419–435, 2002.

[39] A. O. Hero, B. Ma, O. Michel, and J. Gorman, "Alpha-divergence for classification, indexing and retrieval," Tech. Rep. CSPL-328, Communication and Signal Processing Laboratory, University of Michigan, 2001.

[40] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. Springer, 1986.

[41] M. Noshad, K. R. Moon, S. Y. Sekeh, and A. O. Hero, "Direct estimation of information divergence using nearest neighbor ratios," *2017 IEEE International Symposium on Information Theory (ISIT)*, 2017.

[42] C. Zhong, M. Malinen, D. Miao, and P. Fränti, "A fast minimum spanning tree algorithm based on K-means," *Information Sciences*, vol. 295, pp. 1–17, 2015.

[43] M. E. Celebi, H. A. Kingravi, and P. A. Vela, "A comparative study of efficient initialization methods for the k-means clustering algorithm," *Expert Systems with Applications*, vol. 40, no. 1, pp. 200–210, 2013.

[44] H. Steinhaus, "Sur la division des corps matériels en parties," *Bulletin de l'Académie Polonaise des Sciences*, vol. IV, no. 12, pp. 801–804, 1956.

[45] J. Scott, *Multivariate Density Estimation: Theory, Practice, and Visualization*. Wiley Series in Probability and Statistics, John Wiley & Sons, Inc., 1992.

[46] T. Ishigami and T. Homma, "An importance quantification technique in uncertainty analysis for computer models," in *Proceedings of the First International Symposium on Uncertainty Modeling and Analysis*, pp. 398–403, IEEE, 1990.

[47] I. M. Sobol, "Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates," *Mathematics and Computers in Simulation*, vol. 55, no. 1, pp. 271–280, 2001.

[48] T. Crestaux, O. Le Maître, and J.-M. Martinez, "Polynomial chaos expansion for sensitivity analysis," *Reliability Engineering & System Safety*, vol. 94, no. 7, pp. 1161–1172, 2009.

[49] I. G. S. van Hooijdonk, *Plane Couette Flow as Model for the Nocturnal Boundary Layer*. PhD thesis, Technische Universiteit Eindhoven, 2017.

[50] R. W. Walters and L. Huyse, "Uncertainty analysis for fluid mechanics with applications," Tech. Rep. 2002-1, ICASE, 2002.

[51] J. A. S. Witteveen and H. Bijl, "Efficient quantification of the effect of uncertainties in advection-diffusion problems using polynomial chaos," *Numerical Heat Transfer, Part B: Fundamentals*, vol. 53, no. 5, pp. 437–465, 2008.

[52] J. A. S. Witteveen, S. Sarkar, and H. Bijl, "Modeling physical uncertainties in dynamic stall induced fluid–structure interaction of turbine blades using arbitrary polynomial chaos," *Computers & Structures*, vol. 85, no. 11, pp. 866–878, 2007.

[53] B. Yildirim and G. E. Karniadakis, "Stochastic simulations of ocean waves: an uncertainty quantification study," *Ocean Modelling*, vol. 86, pp. 15–35, 2015.

[54] D. Xiu and G. E. Karniadakis, "The Wiener–Askey polynomial chaos for stochastic differential equations," *SIAM Journal on Scientific Computing*, vol. 24, no. 2, pp. 619–644, 2002.

[55] D. Xiu and J. S. Hesthaven, "High-order collocation methods for differential equations with random inputs," *SIAM Journal on Scientific Computing*, vol. 27, no. 3, pp. 1118–1139, 2005.

[56] R. G. Ghanem and P. D. Spanos, *Stochastic Finite Elements: A Spectral Approach*. Dover, 2003.

[57] M. S. Eldred and J. Burkardt, "Comparison of non-intrusive polynomial chaos and stochastic collocation methods for uncertainty quantification," *AIAA paper 2009-976*, 2009.

[58] M. Navarro, J. A. S. Witteveen, and J. Blom, "Stochastic collocation for correlated inputs," in *UNCECOMP 2015*, 2015.

[59] S. A. Smolyak, "Quadrature and interpolation formulas for tensor products of certain classes of functions," *Sov. Math. Dokl.*, vol. 4, pp. 240–243, 1963.

[60] T. Gerstner and M. Griebel, "Numerical integration using sparse grids," *Numerical Algorithms*, vol. 18, no. 3-4, pp. 209–232, 1998.

[61] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, pp. 281–297, 1967.

[62] C. Elkan, "Using the triangle inequality to accelerate k-means," in *Proceedings of the International Machine Learning Conference*, vol. 3, pp. 147–153, 2003.

[63] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern Recognition*, vol. 36, no. 2, pp. 451–461, 2003.

[64] A. M. Bagirov, "Modified global k-means algorithm for minimum sum-of-squares clustering problems," *Pattern Recognition*, vol. 41, no. 10, pp. 3192–3199, 2008.

[65] P. Hansen, E. Ngai, B. K. Cheung, and N. Mladenovic, "Analysis of global k-means, an incremental heuristic for minimum sum-of-squares clustering," *Journal of Classification*, vol. 22, no. 2, pp. 287–310, 2005.

[66] M. B. Cohen, S. Elder, C. Musco, C. Musco, and M. Persu, "Dimensionality reduction for k-means clustering and low rank approximation," in *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pp. 163–172, ACM, 2015.

[67] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1027–1035, SIAM, 2007.

[68] C. Ding and X. He, "K-means clustering via principal component analysis," in *Proceedings of the Twenty-First International Conference on Machine Learning*, pp. 29–36, ACM, 2004.

[69] T. Su and J. Dy, "A deterministic method for initializing k-means clustering," in *Proceedings of the International Conference on Tools with Artificial Intelligence*, pp. 784–786, IEEE, 2004.

[70] A. Guénoche, P. Hansen, and B. Jaumard, "Efficient algorithms for divisive hierarchical clustering with the diameter criterion," *Journal of Classification*, vol. 8, no. 1, pp. 5–30, 1991.

[71] H. Fang and Y. Saad, "Farthest centroids divisive clustering," in *Proceedings of the International Conference on Machine Learning and Applications*, pp. 232–238, IEEE, 2008.

[72] A. Genz, "Testing multidimensional integration routines," in *Proceedings of International Conference on Tools, Methods and Languages for Scientific and Engineering Computation*, pp. 81–94, Elsevier, 1984.

[73] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[74] G. E. P. Box and N. R. Draper, "A basis for the selection of a response surface design," *Journal of the American Statistical Association*, vol. 54, no. 287, pp. 622–654, 1959.

[75] M. Powell, "Radial basis function methods for interpolation to functions of many variables," in *HERCMA*, pp. 2–24, 2001.

[76] C. C. Aggarwal, *Neural Networks and Deep Learning*. Springer, 2018.

[77] D. G. Krige, "A statistical approach to some basic mine valuation problems on the Witwatersrand," *Journal of the Southern African Institute of Mining and Metallurgy*, vol. 52, no. 6, pp. 119–139, 1951.

[78] S. Razavi, B. A. Tolson, and D. H. Burn, "Numerical assessment of metamodelling strategies in computationally intensive optimization," *Environmental Modelling & Software*, vol. 34, pp. 67–86, 2012.

[79] A. C. Davison, *Statistical Models*. Cambridge University Press, 2003.

[80] C. E. Rasmussen and C. K. Williams, *Gaussian Processes for Machine Learning*. MIT press, 2006.

[81] Y. B. Lim, J. Sacks, W. J. Studden, and W. J. Welch, "Design and analysis of computer experiments when the output is highly correlated over the input space," *Canadian Journal of Statistics*, vol. 30, no. 1, pp. 109–126, 2002.

[82] W. De Mulder, G. Molenberghs, G. Verbeke, B. Rengs, and T. Fent, "A comparison of some simple and complex surrogate models: make everything as simple as possible?," in *Proceedings of the Eighth International Conference on Advances in System Simulation*, pp. 26–32, 2016.

[83] H. Chen, J. L. Loeppky, J. Sacks, and W. J. Welch, "Analysis methods for computer experiments: how to assess and what counts?," *Statistical Science*, vol. 31, no. 1, pp. 40–60, 2016.

[84] R. von Mises, *Mathematical Theory of Probability and Statistics*. Academic Press, 1964.

[85] H. Wendland, "Error estimates for interpolation by compactly supported radial basis functions of minimal degree," *Journal of Approximation Theory*, vol. 93, no. 2, pp. 258–272, 1998.

[86] B. Matérn, *Spatial Variation*. PhD thesis, Stockholm University, 1960.

[87] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables*, vol. 55. Courier Corporation, 1964. 10th reprint, 1972.

[88] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[89] A. Sóbester, S. J. Leary, and A. J. Keane, "On the design of optimization strategies based on global response surface approximation models," *Journal of Global Optimization*, vol. 33, no. 1, pp. 31–59, 2005.

[90] A.-L. Cholesky, "Sur la résolution numérique des systèmes déquations linéaires," *Bulletin de la SABIX. Société des amis de la Bibliothèque et de l'Histoire de l'école polytechnique*, no. 39, pp. 81–95, 2005.

[91] M. N. Gibbs, *Bayesian Gaussian Processes for Regression and Classification*. PhD thesis, University of Cambridge, 1997.

[92] W. W. Hager, "Updating the inverse of a matrix," *SIAM Review*, vol. 31, no. 2, pp. 221–239, 1989.

[93] K.-F. Chang, "Strictly positive definite functions," *Journal of Approximation Theory*, vol. 87, no. 2, pp. 148–158, 1996.

[94] A. Saltelli, "Global sensitivity analysis: an introduction," in *Proc. 4th International Conference on Sensitivity Analysis of Model Output (SAMO04)*, pp. 27–43, 2004.

[95] J. E. Oakley and A. O'Hagan, "Probabilistic sensitivity analysis of complex models: a Bayesian approach," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 66, no. 3, pp. 751–769, 2004.

[96] A. Saltelli, P. Annoni, I. Azzini, F. Campolongo, M. Ratto, and S. Tarantola, "Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index," *Computer Physics Communications*, vol. 181, no. 2, pp. 259–270, 2010.

[97] I. M. Sobol, "Sensitivity estimates for nonlinear mathematical models," *Mathematical Modeling & Computational Experiments*, vol. 1, no. 4, pp. 407–414, 1993.

[98] E. Borgonovo, "A new uncertainty importance measure," *Reliability Engineering & System Safety*, vol. 92, no. 6, pp. 771–784, 2007.

[99] K. Blix, "Sensitivity Analysis of Gaussian Process Machine Learning for Chlorophyll Prediction from Optical Remote Sensing," Master's thesis, UiT Norges arktiske universitet, 2014.

[100] I. Csiszár and P. C. Shields, "Information theory and statistics: a tutorial," *Foundations and Trends® in Communications and Information Theory*, vol. 1, no. 4, pp. 417–528, 2004.

[101] S. Da Veiga, "Global sensitivity analysis with dependence measures," *Journal of Statistical Computation and Simulation*, vol. 85, no. 7, pp. 1283–1305, 2014.

[102] B. Krzykacz-Hausmann, "Epistemic sensitivity analysis based on the concept of entropy," in *International symposium on sensitivity analysis of model output*, pp. 53–57, 2001.

[103] S. Rahman, "The f-sensitivity index," *SIAM/ASA Journal on Uncertainty Quantification*, vol. 4, no. 1, pp. 130–162, 2016.

[104] A. F. Cortesi, G. Jannoun, and P. M. Congedo, "Kriging-sparse polynomial dimensional decomposition surrogate model with adaptive refinement," *Journal of Computational Physics*, vol. 380, pp. 212–242, 2019.

[105] E. Borgonovo, G. B. Hazen, and E. Plischke, "A common rationale for global sensitivity measures and their estimation," *Risk Analysis*, vol. 36, no. 10, pp. 1871–1895, 2016.

[106] A. Marrel, B. Iooss, B. Laurent, and O. Roustant, "Calculations of Sobol indices for the Gaussian process metamodel," *Reliability Engineering & System Safety*, vol. 94, no. 3, pp. 742–751, 2009.

[107] J. Svenson, T. Santner, A. Dean, and H. Moon, "Estimating sensitivity indices based on Gaussian process metamodels with compactly supported correlation functions," *Journal of Statistical Planning and Inference*, vol. 144, pp. 160–172, 2014.

[108] M. De Lozzo and A. Marrel, "Estimation of the derivative-based global sensitivity measures using a Gaussian process metamodel," *SIAM/ASA Journal on Uncertainty Quantification*, vol. 4, no. 1, pp. 708–738, 2016.

[109] T. Paananen, J. Piironen, M. R. Andersen, and A. Vehtari, "Variable selection for Gaussian processes via sensitivity analysis of the posterior predictive distribution," *arXiv preprint arXiv:1712.08048v2*, 2018.

[110] E. Borgonovo, W. Castaings, and S. Tarantola, "Model emulation and moment-independent sensitivity analysis: An application to environmental modelling," *Environmental Modelling & Software*, vol. 34, pp. 105–115, 2012.

[111] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Physical Review E*, vol. 69, no. 6, 2004.

[112] S. Rahman, "A polynomial dimensional decomposition for stochastic computing," *International Journal for Numerical Methods in Engineering*, vol. 76, no. 13, pp. 2091–2116, 2008.

[113] A. B. Owen, "Orthogonal arrays for computer experiments, integration and visualization," *Statistica Sinica*, pp. 439–452, 1992.

[114] M. Johnson, L. Moore, and D. Ylvisaker, "Minimax and maximin distance designs," *Journal of Statistical Planning and Inference*, vol. 26, no. 2, pp. 131–148, 1990.

[115] H. Niederreiter, *Random Number Generation and Quasi-Monte Carlo Methods*. Society for Industrial and Applied Mathematics, 1992.

[116] S. Joe and F. Y. Kuo, "Constructing Sobol sequences with better two-dimensional projections," *SIAM Journal on Scientific Computing*, vol. 30, no. 5, pp. 2635–2654, 2008.

[117] V. Dubourg, *Adaptive surrogate models for reliability analysis and reliability-based design optimization*. PhD thesis, Université Blaise Pascal-Clermont-Ferrand II, 2011.

[118] Q. Xu, E. Wehrle, and H. Baier, "Adaptive surrogate-based design optimization with expected improvement used as infill criterion," *Optimization*, vol. 61, no. 6, pp. 661–684, 2012.

[119] M. Soilahoudine, C. Gogu, and C. Bes, "Accelerated adaptive surrogate-based optimization through reduced-order modeling," *AIAA Journal*, vol. 55, no. 5, pp. 1681–1694, 2017.

[120] Z. Wang and M. Ierapetritou, "A novel feasibility analysis method for black-box processes using a radial basis function adaptive sampling approach," *AIChE Journal*, vol. 63, no. 2, pp. 532–550, 2016.

[121] M. Diez, S. Volpi, A. Serani, F. Stern, and E. F. Campana, "Simulation-based design optimization by sequential multi-criterion adaptive sampling and dynamic radial basis functions," *Advances in Evolutionary and Deterministic Methods for Design, Optimization and Control in Engineering and Sciences*, pp. 213–228, 2018.

[122] B. Gaspar, A. Teixeira, and C. Guedes Soares, "Adaptive surrogate model with active refinement combining Kriging and a trust region method," *Reliability Engineering & System Safety*, vol. 165, pp. 277–291, 2017.

[123] P. Moonen and J. Allegrini, "Employing statistical model emulation as a surrogate for CFD," *Environmental Modelling & Software*, vol. 72, pp. 77–91, 2015.

[124] A. Golzari, M. Haghighat Sefat, and S. Jamshidi, "Development of an adaptive surrogate model for production optimization," *Journal of Petroleum Science and Engineering*, vol. 133, pp. 677–688, 2015.

[125] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. Kevin Tucker, "Surrogate-based analysis and optimization," *Progress in Aerospace Sciences*, vol. 41, pp. 1–28, Jan 2005.

[126] H. Liu, Y.-S. Ong, and J. Cai, "A survey of adaptive sampling for global metamodeling in support of simulation-based complex engineering design," *Structural and Multidisciplinary Optimization*, vol. 57, no. 1, pp. 393–416, 2017.

[127] L. Jensen, C. Mørch, P. Sørensen, and K. Svendsen, "Wake measurements from the Horns Rev wind farm," in *European wind energy conference*, vol. 9, 2004.

[128] "Horns Rev 1." `https://powerplants.vattenfall.com/en/horns-rev`. Accessed: 2018-02-09.

[129] B. Hu, "Design of a Simple Wake Model for the Wind Farm Layout Optimization Considering the Wake Meandering Effect," Master's thesis, Technische Universiteit Delft, 2016.

[130] M. Bastankhah and F. Porté-Agel, "A new analytical model for wind-turbine wakes," *Renewable Energy*, vol. 70, pp. 116–123, 2014.

[131] S. Frandsen, R. Barthelmie, S. Pryor, O. Rathmann, S. Larsen, J. Højstrup, and M. Thøgersen, "Analytical modelling of wind speed deficit in large offshore wind farms," *Wind Energy*, vol. 9, no. 1-2, pp. 39–53, 2006.

[132] A. Niayifar and F. Porté-Agel, "A new analytical model for wind farm power prediction," *Journal of Physics: Conference Series*, vol. 625, article no. 012039, 2015.

[133] R. J. Barthelmie, S. T. Frandsen, M. N. Nielsen, S. C. Pryor, P.-E. Rethore, and H. E. Jørgensen, "Modelling and measurements of power losses and turbulence intensity in wind turbine wakes at Middelgrunden offshore wind farm," *Wind Energy*, vol. 10, no. 6, pp. 517–528, 2007.

[134] A. Crespo and J. Hernández, "Turbulence characteristics in wind-turbine wakes," *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 61, no. 1, pp. 71–85, 1996.

[135] J. Choi and M. Shan, "Advancement of Jensen (Park) wake model," in *European Wind Energy Conference and Exhibition 2013*, vol. 3, pp. 1790–1797, European Wind Energy Association, 2013.

[136] C. Jung and D. Schindler, "On the inter-annual variability of wind energy generation – A case study from Germany," *Applied Energy*, vol. 230, pp. 845–854, 2018.

[137] B. Alonzo, H.-K. Ringkjob, B. Jourdier, P. Drobinski, R. Plougonven, and P. Tankov, "Modelling the variability of the wind energy resource on monthly and seasonal timescales," *Renewable Energy*, vol. 113, pp. 1434–1446, 2017.

[138] P. Richter, J. Wolters, R. Çakar, A. Verhoeven-Mrosek, and M. Frank, "Uncertainty quantification of offshore wind farms," 2018. Preprint on ResearchGate.

[139] E. Lundtang Petersen, I. Troen, H. Ejsing Jørgensen, and J. Mann, "The new European wind atlas," *Energy Bulletin*, no. 17, pp. 34–39, 2014.

[140] Copernicus Climate Change Service (C3S), "ERA5: fifth generation of ECMWF atmospheric reanalyses of the global climate." `https://cds.climate.copernicus.eu/cdsapp#!/home`, 2017. Accessed: 2019-04-15.

[141] A. Berkers, "Higher-order quadrature rules for data sets." Bachelor's thesis, Eindhoven University of Technology, 2018.